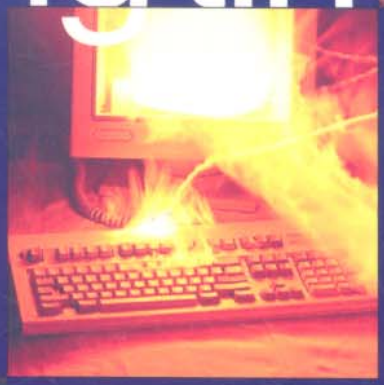
NGUYỄN VĂN BA

Phân tích và thiết kế hệ thống thông tin



CÁC PHƯƠNG PHÁP CÓ CẤU TRÚC

NHÀ XUẤT BẢN ĐẠI HỌC QUỐC GIA HÀ NỘI

NGUYỄN VĂN BA

Phân tích và thiết kế hệ thống thông tin

(In lần thứ tư)

Sách dùng cho:

- Sinh viên các trường Đại học, Cao đẳng
- Các nhà xây dựng hệ thống chuyên nghiệp
- Các kỹ sư phân tích và thiết kế

NHÀ XUẤT BẢN ĐẠI HỌC QUỐC GIA HÀ NỘI

Chịu trách nhiệm xuất bản:

Giám đốc: PHÙNG QUỐC BẢO
Tổng biên tập: PHẠM THÀNH HƯNG

Biên tập: NGUYỄN VĂN BA
HỒ ĐỒNG

Trình bày bìa: VĂN SÁNG

PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG THÔNG TIN

Mã số: 2L - 50 ĐH2006
In 1000 cuốn, khổ 16 x 24 tại Trung tâm in tranh Tuyên truyền Cổ động
Số xuất bản: 85 - 2006/CXB/70 - 01/ĐHQGHN, ngày 24/01/2006
Quyết định xuất bản số: 384 LK/XB
In xong và nộp lưu chiểu quý IV năm 2006

Lời nói đầu

Cuốn sách này đề cập tới việc phân tích và thiết kế một hệ thống thông tin (HTT), nhấn mạnh tới HTT quản lý. Đó là kết quả của một giáo trình giảng dạy ở bậc đại học cho ngành Công nghệ Thông tin. Giáo trình này thường được giảng ở năm thứ tư, sau khi sinh viên đã học qua một số ngôn ngữ và phương pháp lập trình.

Phân tích và thiết kế hệ thống nhằm mục đích gì?

HTT nào cũng có một đời sống, từ lúc khai sinh đến lúc bị phế bỏ. Đó là một quá trình trải qua một số giai đoạn nhất định. Các giai đoạn chính thường là: tìm hiểu nhu cầu, phân tích, thiết kế, cài đặt, khai thác và bảo dưỡng. Không nhất thiết các giai đoạn đó được đi qua một cách tuyến tính, mà có thể vòng đi vòng lại nhiều lần. Vì vậy đời sống của HTT thường được gọi là chu trình sống hay chu trình phát triển.

- Giai đoạn tìm hiểu nhu cầu là nhằm làm rõ HTT sẽ được lập ra phải đáp ứng các nhu cầu gì của người dùng - các nhu cầu trước mắt và tương lai, nhu cầu tường minh và tiềm ẩn.
- Giai đoạn phân tích là nhằm đi sâu vào bản chất và chi tiết của hệ thống, cho thấy là hệ thống phải thực hiện những việc gì và các dữ liệu mà nó đề cập là những dữ liệu nào, có cấu trúc ra sao.
- Giai đoạn thiết kế là nhằm đưa ra các quyết định về cài đặt hệ thống, để sao cho hệ thống thỏa mãn được các yêu cầu mà giai đoạn phân tích đã đưa ra, đồng thời lại thích ứng với các điều kiện ràng buộc trong thực tế.
- Giai đoạn cài đặt bao gồm hai công việc chính, là lập trình và kiểm định, nói theo thuật ngữ của ngành xây dựng, thì đó là giai đoạn thi công nhằm chuyển các kết quả phân tích và thiết kế trên giấy, thành một hệ thống chạy được.
- Giai đoạn khai thác và bảo dưỡng là giai đoạn đưa hệ thống vào sử dụng, đồng thời thực hiện các chỉnh sửa, khi phát hiện thấy hệ thống còn có chỗ chưa thích hợp.

Điều đáng nói là dù chu trình sống là tuyến tính hay lặp, thì vẫn có một nguyên tắc bất di bất dịch là: Cài đặt chỉ được thực hiện sau khi đã phân tích

và thiết kế, Điều đáng tiếc là có không ít người, kể cả một số người lập trình có kinh nghiệm, lại xem thường khâu phân tích và thiết kế, thậm chí là vẫn thường xuyên lập trình... chay, nghĩa là lập trình mà không hề phân tích và thiết kế. Chẳng phải là cách làm này đã không từng thành công với một số hệ thống nhỏ, đơn giản và có ít người tham gia xây dựng (thường chỉ là một hay hai người). Còn đối với các hệ thống lớn và phức tạp, có nhiều người tham gia xây dựng (trên mười người), thì thiếu phân tích và thiết kế khó có thể đạt đến một hệ thống có chất lượng cao, thậm chí khó có thể có sự nhất trí cao trọng tập thể các người xây dựng hệ thống.

Có người lại lập luận rằng: nguyên tắc trên là đúng cho những năm 70, 80 của thế kỷ vừa qua. Lúc đó lập trình là một công việc quá nặng nhọc, tốn thời gian và tốn tiền. Do đó phải phân tích và thiết kế kỹ để giảm bớt gánh nặng cho lập trình. Còn ngày nay với công nghệ thế hệ bốn 4GT (fourth generation technology) bao gồm nhiều cái mới như các ngôn ngữ khai báo, các pakét ứng dụng, các phần mềm làm giao diện, các công cụ trợ giúp CASE v.v... Việc lập trình trở nên nhanh và rẻ hơn rất nhiều (còn khoảng 10% giá thành và 5% thời gian so với lập trình truyền thống), phải chăng lúc này phân tích và thiết kế lại không còn cần nữa? Dường như là đúng thế, bởi cách thực hiện hệ thống với 4GT là: lợi dụng khả năng lập trình nhanh, lập một phương án thô của hệ thống (một nguyên mẫu), đem cho người dùng dùng thử. Phát hiện được chỗ người dùng chưa bằng lòng, thì chỉnh sửa để lại đưa ra một phương án mới. Cứ thế, thành lập dãy các nguyên mẫu, rốt cục người ta đạt được hệ thống thỏa mãn mọi yêu cầu của người dùng.

Thực ra là không đúng, bởi vì hai lẽ:

Người dùng chấp nhận không có nghĩa là hệ thống đáp ứng đúng các nhu cầu đặt ra đối với nó. Bởi vì người dùng vốn quen dùng hệ thống cũ lạc hậu, nay thấy cái mới rất dễ thỏa mãn. Vả lại người dùng vốn không ý thức được nhu cầu của mình một cách chính xác. Vậy người xây dựng hệ thống phải thẩm định nhu cầu của người dùng, phân tích nó thì mới rõ được là đó có phải là nhu cầu chính đáng hay không.

Hơn nữa, những cái mới đưa vào cho mỗi bước xây dựng nguyên mẫu không thể được chọn lựa một cách tùy tiện, hú họa được. Ngược lại phải bảo đảm đó là giải pháp tốt, hợp lý và hữu hiệu. Có thế thì nguyên mẫu mới phát triển đúng hướng và tiệm cận dần đến hệ thống mong muốn. Và như vậy cũng không tránh khỏi phải phân tích và thiết kế về các yếu tố mới bổ sung này.

Tóm lại là làm nguyên mẫu (với 4GT) thì vẫn phải tiến hành phân tích và thiết kế trong từng vòng lặp, dù rằng đấy thường là bước phân tích thiết kế không hoàn chỉnh và làm càng nhanh càng tốt.

Phân tích và thiết kế có cần có phương pháp?

Có việc gì làm một cách nghiêm túc mà lại không cần áp dụng một phương pháp? (kể cả làm thơ!). Huống chi việc phân tích và thiết kế hệ thống vốn là một việc phức tạp và trường kỳ, đương nhiên là rất cần được triển khai theo một phương pháp hợp lý.

Vậy trước hết phương pháp phân tích và thiết kế là gì? Và nó giúp gì cho người xây dựng hệ thống? Một phương pháp phân tích và thiết kế là sự hợp thành của ba yếu tố:

- Một tập hợp các khái niệm và mô hình, bao gồm các khái niệm cơ bản sử dụng trong phương pháp cùng với các cách biểu diễn chúng (thường dưới dạng đồ thị).
- Một tiến độ triển khai, bao gồm các bước đi lần lượt, các hoạt động cần làm.
- Một công cụ trợ giúp, là một phần mềm giúp cho việc triển khai hệ thống thực hiện theo phương pháp được chặt chẽ và nhanh chóng.

Trước những năm 60, chưa định hình những phương pháp rõ rệt cho phân tích và thiết kế hệ thống. Người ta xây dựng hệ thống một cách tùy tiện, theo sở thích và kinh nghiệm cá nhân. Kết quả phân tích là những tập tài liệu dày cộm, trình bày dưới dạng văn tự dông dài, khó đọc và khó trao đổi. Từ những năm 70 tới nay, nhiều phương pháp phân tích và thiết kế lần lượt ra đời. Mỗi phương pháp đều có ưu điểm và nhược điểm riêng, có thể được ưa chuộng ở nơi này, nhưng lại được ít ưa chuộng ở nơi khác. Sự phong phú và đa dạng trong phương pháp như vậy cũng có nghĩa là sự không thống nhất, không chuẩn hóa.

Tuy nhiên, trải qua thời gian, một số phương pháp đã tỏ ra là có một sức sống dẻo dai, bám trụ được cho đến tận hôm nay. Trong số này phải kể trước hết các phương pháp được gọi dưới một cái tên chung là các phương pháp có cấu trúc (hay các phương pháp trên xuống). Cũng không thể không kể đến một trào lưu mới, mãnh liệt: đó là sự ra đời khá ồ ạt, từ năm 1990, của các phương pháp phân tích và thiết kế hướng đối tượng, để rồi quy vào một cái chuẩn, xuất hiện năm 1997, là UML (Unified Modeling Language).

Vậy cuốn sách này trình bày phân tích và thiết kế theo phương pháp nào?

Đúng là trong một cuốn sách, ta không thể trình bày tất cả các phương pháp được, mà chỉ có thể chọn lấy một phương pháp để trình bày. Ở đây, chúng tôi chọn phương pháp phân tích và thiết kế có cấu trúc, vì các lẽ sau:

- Phương pháp có cấu trúc, trải qua thời gian đã chứng tỏ được tính kinh điển của nó. Học nó trước hết là học cách tư duy nhất quán và chặt chẽ của nó. Điều này là hết sức cần thiết cho một giáo trình "Nhập môn", lần đầu đưa người đọc vào một thế giới mới như giáo trình này.

- Phương pháp có cấu trúc là phương pháp dung dị, không cầu kỳ như một số phương pháp khác, dễ áp dụng, nhưng lại rất hữu hiệu. Ngày nay nó chưa lạc hậu, mà vẫn còn phát huy tác dụng tốt. Bằng chứng là một hệ thống lớn và hiện đại như ORACLE vẫn tiếp tục sử dụng nó.

Thực ra, như đã nói ở trên, dưới cái tên phương pháp có cấu trúc có nhiều phương pháp của các tác giả hay nhóm tác giả khác nhau. Các phương pháp này nhất quán trong cùng một phương hướng tư duy (có cấu trúc và trên xuống), nhưng mỗi phương pháp lại chỉ đề cập một phương diện của quá trình phân tích và thiết kế, cho nên người ta thường sử dụng chúng liên hoàn với nhau, bổ sung cho nhau trong cùng một đề án triển khai hệ thống. Cuốn sách sẽ lần lượt trình bày các phương pháp này dọc theo các giai đoạn của chu trình phát triển hệ thống.

Cuốn sách, như đã nói ở trên, là nhằm trước hết làm tài liệu tham khảo cho sinh viên ngành Công nghệ Thông tin các trường Đại học Bách khoa, Đại học Quốc gia. Nó cũng thích hợp với các hệ đào tạo chính qui, phi chính qui và từ xa ở Viện Đại học mở. Ngoài ra nó cũng có thể giúp ích cho các người đang thực hiện một đề án triển khai hệ thống dựa theo phương pháp phân tích và thiết kế có cấu trúc. Và nó cũng có thể có ích cho các thầy giáo trong lĩnh vực này, như là một nguồn tài liệu để soạn bài giảng.

Cuối cùng tác giả xin tỏ lòng cám ơn các đồng nghiệp ở khoa CNTT ĐHBK Hà nội, đã hối thúc và giúp đỡ tác giả nhiều trong việc biên soạn cuốn sách này. Mọi ý kiến về cuốn sách, xin được trao đổi với tác giả qua địa chỉ điện tử: banv@it-hut.edu.vn

NGUYỄN VĂN BA
(ĐHBK Hà nội)

Chương I
ĐẠI CƯƠNG VỀ HỆ THỐNG VÀ CHU TRÌNH PHÁT TRIỂN HỆ THỐNG

Chương mở đầu này sẽ trình bày một cách khái quát về khái niệm hệ thống, một số hệ thống đáng chú ý (hệ thống kinh doanh/ dịch vụ, hệ thống tin học), rồi tiếp đó đề cập các chu trình phát triển của các hệ thống tin học, bao gồm các bước phân tích, thiết kế, ... và cuối cùng là các phương pháp phân tích và thiết kế hệ thống, đặc biệt lưu ý các phương pháp được vận dụng trong tập sách như là SA, E/A, SA-RT, SD.

§1. HỆ THỐNG KINH DOANH/ DỊCH VỤ VÀ HỆ THỐNG THÔNG TIN TRONG NÓ

1. Khái niệm chung về hệ thống

Thuật ngữ hệ thống không phải là mới. Từ lâu người ta đã nói đến hệ thống mặt trời, hệ thống triết học, hệ thống luật pháp, hệ thống thủy lực, hệ thống cơ khí, hệ thống tuần hoàn, hệ thống thông tin...

Một cách đơn giản và vắn tắt nhất, thì ta có thể hiểu: Hệ thống là một tập hợp gồm nhiều phần tử, có các mối quan hệ ràng buộc lẫn nhau và cùng hoạt động hướng tới một mục đích chung.

Ta đề cập sâu thêm một số khía cạnh trong định nghĩa này:

a) Các phần tử của hệ thống

Các phần tử nói đây chẳng qua là các thành phần hợp thành hệ thống, được hiểu theo nghĩa rất rộng rãi:

- Các phần tử đó có thể rất đa dạng: chẳng hạn trong hệ thống mặt trời thì các phần tử là mặt trời, quả đất, hỏa tinh,...; trong hệ thống thần kinh thì các phần tử là bộ óc, tủy sống, các dây thần kinh,...; có khi các phần tử lại là những đối tượng trừu tượng, như là một phương pháp, một lập luận,

một quy tắc... như trong các hệ thống tư tưởng. Như vậy các phần tử có thể là rất khác biệt về bản chất, không những giữa các hệ thống khác nhau, mà có thể ngay trong cùng một hệ thống.

- Các phần tử lại không nhất thiết là đơn giản, sơ đẳng, mà thường khi là những thực thể phức tạp, khiến khi đi sâu vào chúng, ta lại phải xem chúng là các hệ thống. Bởi thế, hệ thống thường có tính phân cấp: hệ thống hợp thành từ nhiều hệ thống con, và trong mỗi hệ thống con đó lại có các hệ thống nhỏ hơn...

b) Các quan hệ giữa các phần tử

Các phần tử của một hệ thống không phải được tập hợp lại một cách ngẫu nhiên, rời rạc, mà giữa chúng luôn tồn tại những quan hệ (hay các mối ràng buộc lẫn nhau), tạo thành một cấu trúc (hay tổ chức). Chẳng hạn, trong một hệ thống hành chính, gồm các cán bộ và nhân viên, thì giữa họ tồn tại các mối ràng buộc về phân cấp, phân quyền, các quan hệ về đoàn thể, các quan hệ về dân sự v.v...

Cần phân biệt:

- Các quan hệ ổn định, tồn tại lâu dài, ví dụ A là thủ trưởng của B;
- Các quan hệ bất thường, tạm thời, ví dụ A và B vừa được cử đi công tác cùng nhau.

Khi xem xét tính tổ chức của một hệ thống, đương nhiên ta phải đề cập trước hết đến các quan hệ ổn định, lâu dài.

Tuy nhiên nói ổn định, không nhất thiết phải hiểu là hoàn toàn bất biến, tĩnh tại. Trái lại phần lớn các hệ thống đáng quan tâm đều có tính biến động. Biến động song vẫn giữ sự ổn định trong tổ chức, trong các quan hệ giữa các phần tử, nghĩa là vẫn giữ cái bản chất, hay các đặc trưng cốt lõi của hệ thống.

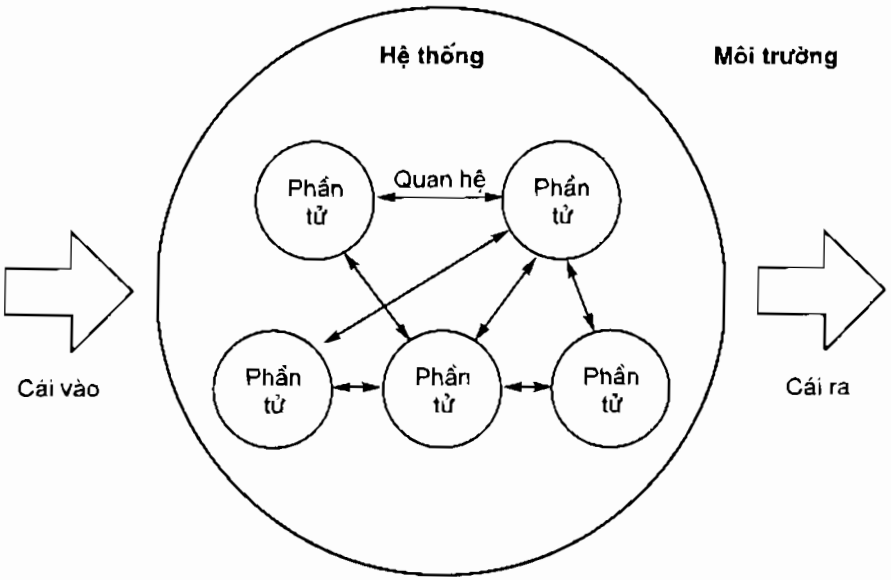
c) Sự hoạt động và mục đích của hệ thống

Sự biến động của hệ thống thể hiện trên hai mặt:

- Sự *tiến triển*, tức là các thành phần của nó (các phần tử và các quan hệ) có thể có phát sinh, có tăng trưởng, có suy thoái, có mất đi.
- Sự *hoạt động*, tức là các phần tử của hệ thống, trong các mối ràng buộc đã định, cùng cộng tác với nhau để thực hiện một mục đích chung của hệ thống.

Mục đích của hệ thống thường thể hiện ở chỗ hệ thống nhận những *cái vào* để chế biến thành những *cái ra* nhất định. Chẳng hạn một hệ thống thu hình, nhận vào năng lượng điện cùng các sóng vô tuyến từ đài phát, để biến thành các hình ảnh trên màn hình; còn một hệ thống sản xuất thì nhận vào các nguyên vật liệu, tiền và dịch vụ để sản xuất ra các thành phẩm, hàng hóa.

Khi nói hệ thống nhận cái vào và xuất cái ra, thì đã ngầm hiểu là hệ thống phải ở trong một *môi trường* và nó nhận cái vào từ môi trường đó và xuất cái ra trả lại môi trường đó. Mà môi trường thì chẳng qua lại là một hệ thống nào đó, có giao tiếp với hệ thống đang xét.



Hình I.1 Hình dung một hệ thống

Vì mục đích của cuốn sách, ta quan tâm đặc biệt tới một vài loại hệ thống được đề cập sau đây.

2. Hệ thống kinh doanh/ dịch vụ và các hệ con của nó

a) Hệ thống kinh doanh/ dịch vụ

Là hệ thống mà mục đích là kinh doanh hay dịch vụ.

- Kinh doanh là hoạt động của con người nhằm mang lại *lợi nhuận* (tức là thu giá trị thặng dư). Chẳng hạn sản xuất, phân phối hay lưu thông sản phẩm là các hoạt động kinh doanh.

- Dịch vụ là hoạt động của con người nhằm mang lại *lợi ích* (tức là cung cấp giá trị sử dụng). Chú ý rằng có những dịch vụ là phi lợi nhuận (bởi ở đó không thể có tăng năng suất, để từ đó tạo ra giá trị thặng dư), ví dụ các hoạt động giáo dục, y tế, từ thiện v.v...

Hệ thống kinh doanh/ dịch vụ nói ở đây có thể ở những quy mô khác nhau. Quy mô nhỏ như là một phân xưởng, một cửa hàng. Quy mô vừa như là một nhà máy, một công ty, một bệnh viện, một trường đại học. Quy mô lớn như là một tổng công ty, một ngành sản xuất, một tập đoàn kinh doanh đa quốc gia... Để cho gọn, sau này ta thường gọi hệ thống kinh doanh/ dịch vụ là *doanh nghiệp* hay khái quát hơn là *cơ quan*.

Đặc điểm chung của các hệ thống kinh doanh/ dịch vụ so với các hệ thống khác, như là các hệ thống vật lý, kỹ thuật hay sinh học, là: chúng là của con người và có con người tham gia.

- Của con người, cho nên các mục tiêu của chúng là do con người định ra;
- Có con người tham gia, cho nên con người thường xuyên góp phần thúc đẩy hay kìm hãm sự phát triển của hệ thống, bởi vì con người:
  - có trí thông minh, có khả năng sáng tạo;
  - có tình cảm, có tham vọng.

Đặc điểm chung nói trên dẫn tới hai nét nổi bật của các hệ thống kinh doanh/ dịch vụ:

- Vai trò của *cơ chế điều khiển* (trong kinh doanh thường gọi là sự *quản lý*) là rất quan trọng, nhằm giữ cho hệ thống hướng đúng đích và đạt kết quả với chất lượng cao.
- Vai trò của *thông tin* cũng rất quan trọng, nhằm phục vụ cho nhu cầu giao tiếp, trao đổi giữa con người với nhau.

b) Các hệ thống con trong hệ thống kinh doanh/ dịch vụ

Bởi sự tồn tại của nhiệm vụ quản lý bên cạnh nhiệm vụ sản xuất như đã nói ở trên, cho nên các hệ thống kinh doanh/ dịch vụ luôn bao gồm hai hệ thống con:

- Hệ thống tác nghiệp, gồm các người, phương tiện, phương pháp trực tiếp tham gia vào quá trình biến đổi luồng những cái vào thành luồng những cái ra (thể hiện mục đích kinh doanh hay dịch vụ) của hệ thống.

- Hệ thống quản lý, gồm các người, phương tiện, phương pháp cho phép điều khiển, kiểm soát hoạt động tác nghiệp hướng đúng vào mục đích kinh doanh hay dịch vụ.

Về mặt hình thức thì hoạt động quản lý luôn luôn là một dãy nối tiếp của hai việc:

- đề xuất một quyết định kinh doanh,

- thực thi quyết định kinh doanh.

Ta hiểu *quyết định* là một sự chọn lựa một trong những phương án hành động có thể để giải quyết một vấn đề nào đó.

Quyết định có thể ở nhiều mức (hay tầm quan trọng) khác nhau:

+ Quyết định chiến lược hay kế hoạch, có ảnh hưởng lâu dài và rộng khắp đối với hệ thống;

+ Quyết định chiến thuật hay điều phối, có ảnh hưởng ngắn hạn hay cục bộ đối với hệ thống;

+ Quyết định tác vụ nhằm tổ chức thực hiện từng vụ việc cụ thể trong hoạt động của hệ thống.

Mọi quyết định đều được đề xuất qua hai bước:

- tìm hiểu vấn đề;

- chọn lựa giải pháp.

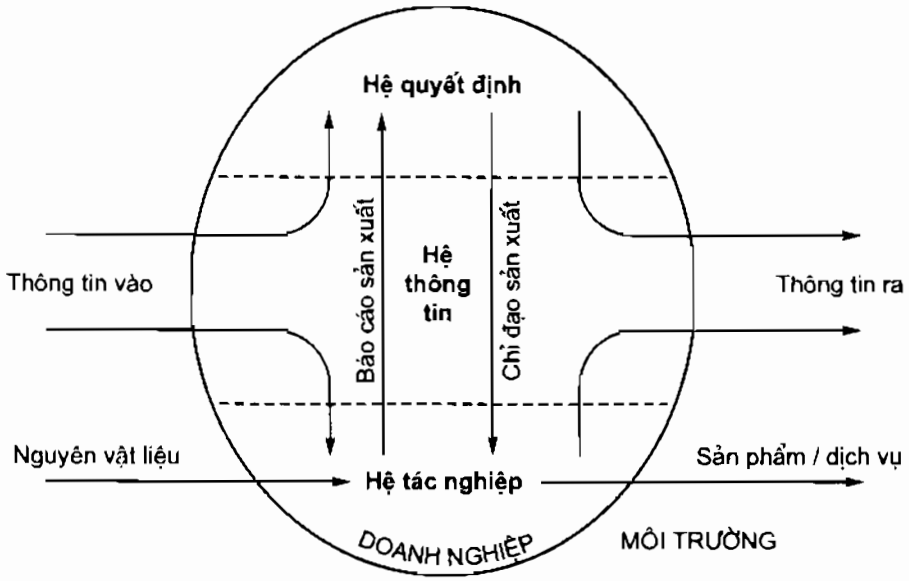
Như vậy, trước khi ra quyết định cần phải thu thập các thông tin liên quan. Thông thường thì các thông tin có ích cho quyết định phải được kết xuất từ nhiều nguồn thông tin phức tạp, thông qua các quá trình thu gom, lưu trữ, xử lý. Sau khi ra quyết định, thì quyết định phải được truyền đạt đến nơi thực hiện, cùng với những thông tin cần thiết cho việc thực thi quyết định đó. Nói thế có nghĩa là, trong quản lý, bên cạnh nhiệm vụ đề xuất các quyết định kinh doanh, luôn có nhiệm vụ xử lý thông tin. Vì vậy hệ thống (con) quản lý trong hệ thống kinh doanh/ dịch vụ lại có thể tách thành hai hệ thống con:

- Hệ quyết định, gồm các người, phương tiện, phương pháp thực hiện việc đề xuất các quyết định kinh doanh;

- Hệ thông tin, gồm các người, phương tiện, phương pháp tham gia vào việc xử lý các thông tin kinh doanh.

Tóm lại, hệ thống kinh doanh/ dịch vụ chứa ba hệ thống con: hệ tác nghiệp, hệ quyết định và hệ thông tin, mà mối liên quan về thông tin giữa

chúng được diễn tả như trong Hình I.2, ở đó thấy rõ vai trò trung gian của hệ thống thông tin trong doanh nghiệp.



Hình I.2. Các phân hệ của HT kinh doanh/ dịch vụ

Chú ý rằng việc phân chia hệ thống kinh doanh/ dịch vụ thành ba hệ thống con như trên chỉ có ý nghĩa phương pháp luận, nhằm cho ta một cách nhìn, một cách nghiên cứu đối với hệ thống, chứ không phải là một sự phân chia về tổ chức. Thực vậy, giữa các hệ thống con đó khó có thể vạch được ranh giới thật rành rẽ: chẳng hạn có thể có người vừa làm công tác lãnh đạo, vừa làm công tác chuyên môn, vừa tham gia xử lý thông tin, và như thế anh ta là thành viên của cả ba hệ thống con. Mặt khác thì trên thực tế, các cơ quan thường được tổ chức thành các phòng, ban, khoa, bộ môn, phân xưởng, cửa hàng v.v... để phân công thực hiện các chức năng quản lý hay tác nghiệp; nhưng các chức năng này lại thường đan xen với nhau trong các bộ phận, mà không chia tách một cách hoàn toàn rành rẽ được giữa các bộ phận đó.

Tuy nhiên, nhìn một cách khái quát vào một doanh nghiệp hay cơ quan, ta luôn thấy:

- Những người mà nhiệm vụ chính là đề xuất các quyết định (các cán bộ lãnh đạo các cấp);
- Những người mà nhiệm vụ chính là xử lý thông tin (nhân viên các phòng ban).

- Những người mà nhiệm vụ chính là trực tiếp sản xuất hay dịch vụ (công nhân, kỹ sư, bác sĩ, thầy giáo...).

Vì thế mà cách tiếp cận hệ thống như trên vẫn là có ích.

c) Thông tin và xử lý thông tin trong doanh nghiệp

Như trên đã nói, nhiệm vụ của hệ thống thông tin trong doanh nghiệp là xử lý các thông tin kinh doanh.

Ta hiểu xử lý thông tin là tập hợp những thao tác áp dụng lên các thông tin nhằm chuyển chúng về một dạng trực tiếp sử dụng được: làm cho chúng trở thành hiểu được, tổng hợp hơn, truyền đạt được, hoặc có dạng đồ họa v.v...

Nói cụ thể hơn thì một xử lý thông tin đề cập một hay một số trong các thao tác cơ bản sau:

- Ghi nhận và lưu trữ một thông tin lên một giá mang;
- Sắp xếp các thông tin theo một trật tự nào đó;
- Tham khảo thông tin (chẳng hạn tham khảo một tệp, một cơ sở dữ liệu, tìm kiếm tư liệu...).
- Điều chỉnh dạng của thông tin;
- Điều chỉnh nội dung thông tin (cập nhật, biến đổi);
- Từ một số thông tin rút ra một thông tin khác (tính toán, kết xuất);
- Chuyển thông tin đi xa (viễn thông);
- Phân phối thông tin tới một người hay một nhóm người (truyền đạt).

Thông tin kinh doanh (tức là thông tin dùng cho mục đích quản lý trong các doanh nghiệp) thường được phân theo hai loại chính:

+ Thông tin tự nhiên: là các thông tin sinh ra và thu nhận bởi con người trực tiếp bằng các cơ quan biểu đạt hay cảm thụ tự nhiên của người. Đó cụ thể là:

- thông tin viết (văn bản),
- thông tin hình ảnh (tranh, ảnh, sơ đồ,...),
- thông tin miệng (lời nói),
- thông tin xúc giác, khứu giác, âm thanh,...

+ Thông tin có cấu trúc (dữ liệu): Là các thông tin được chắt lọc từ các thông tin tự nhiên, bằng cách cấu trúc hóa lại, làm cho cô đọng hơn, chặt chẽ hơn. Các thông tin chứa đựng trong các loại sổ sách, trong các tệp máy tính đều là các thông tin có cấu trúc. Chúng không còn có dáng vẻ tự nhiên nữa và nói chung chúng là các dãy giá trị (số, chữ,...) được bố trí theo một quy cách nào đấy (cú pháp) và được hiểu nghĩa theo một cách giải thích nào đấy (ngữ nghĩa).

Việc sử dụng thông tin có cấu trúc thay vì thông tin tự nhiên mang lại hai điều lợi:

- Nhờ tính cô đọng, ngắn gọn mà thông tin có cấu trúc được truyền đạt nhanh hơn, với độ tin cậy cao hơn, và khi lưu giữ trên giá mang (giấy, vật liệu từ...) chúng chiếm không gian bé hơn;
- Nhờ có cú pháp chặt chẽ, thông tin có cấu trúc cho phép thực hiện các tính toán, các xử lý theo giải thuật: từ một tập hợp các thông tin, có thể nhận được một cách tự động những thông tin mới (thông tin kết xuất).

Nói chung thì cả hai loại thông tin tự nhiên và có cấu trúc đều tham gia vào quá trình quản lý, song các thông tin tự nhiên chủ yếu chỉ bó gọn trong công tác văn thư, còn các mặt hoạt động quan trọng của quản lý, như quản lý tài chính, nhân sự, thiết bị, khách hàng v.v... thì chủ yếu lại sử dụng các thông tin có cấu trúc. Vì vậy mà sau này ta lưu ý nhiều đến các thông tin có cấu trúc mà ít nhắc đến các thông tin tự nhiên.

d) Hai thành phần cơ bản của hệ thống thông tin

Nếu không kể con người và thiết bị, thì hệ thống thông tin trong doanh nghiệp có hai thành phần cơ bản: Các dữ liệu ghi nhận thực trạng của doanh nghiệp và các xử lý cho phép biến đổi các dữ liệu.

- *Các dữ liệu*: Đó là các thông tin được lưu và duy trì nhằm phản ánh thực trạng hiện thời hay quá khứ của doanh nghiệp. Có thể tách các dữ liệu này thành hai phần:
    + Các dữ liệu phản ánh cấu trúc nội bộ của cơ quan, như dữ liệu về nhân sự, nhà xưởng, thiết bị v.v... Cấu trúc cơ quan không phải là cố định, mà có thể có biến động khi có *một sự kiện tiến hóa* xảy ra (chẳng hạn khi một nhân viên chết, một thiết bị mới được bổ sung...). Sự kiện tiến hóa thường xảy ra bất chợt, ngoài ý muốn của con người. Sự điều chỉnh lại các dữ liệu cho thích hợp khi có sự kiện tiến hóa xảy ra gọi là sự *cập nhật*.

+ Các dữ liệu phản ánh các hoạt động kinh doanh/ dịch vụ của cơ quan, như là dữ liệu về sản xuất, mua bán, giao dịch v.v... Hoạt động kinh doanh/ dịch vụ nhằm biến đổi luồng vào/ ra của doanh nghiệp có thể xem là sự tiếp nối của hàng loạt các sự việc sơ đẳng, gọi là các *sự kiện hoạt động* (chẳng hạn nhận một lô hàng, hoàn thành một mẻ sản phẩm, một đơn hàng tới, thanh toán một hóa đơn v.v...). Khi có một sự kiện hoạt động xảy ra thì phải *ghi nhận* (hay thu thập) nó, và như vậy làm thay đổi các dữ liệu phản ánh các hoạt động kinh doanh/ dịch vụ của doanh nghiệp.

- *Các xử lý*: Đó là những quá trình biến đổi thông tin, nhằm vào hai mục đích chính:

  + Sản sinh các thông tin theo thể thức quy định, chẳng hạn các chứng từ giao dịch (đơn mua hàng, hóa đơn...), các báo cáo, các bản thống kê v.v...

  + Trợ giúp cho các quyết định, thông thường là cung cấp những thông tin cần thiết cho việc chọn lựa một quyết định của lãnh đạo, nhưng cũng có thể là thực hiện một sự chọn lựa quyết định (một cách tự động), nếu đó là loại quyết định dựa trên giải thuật (khác với loại quyết định dựa trên trực quan).

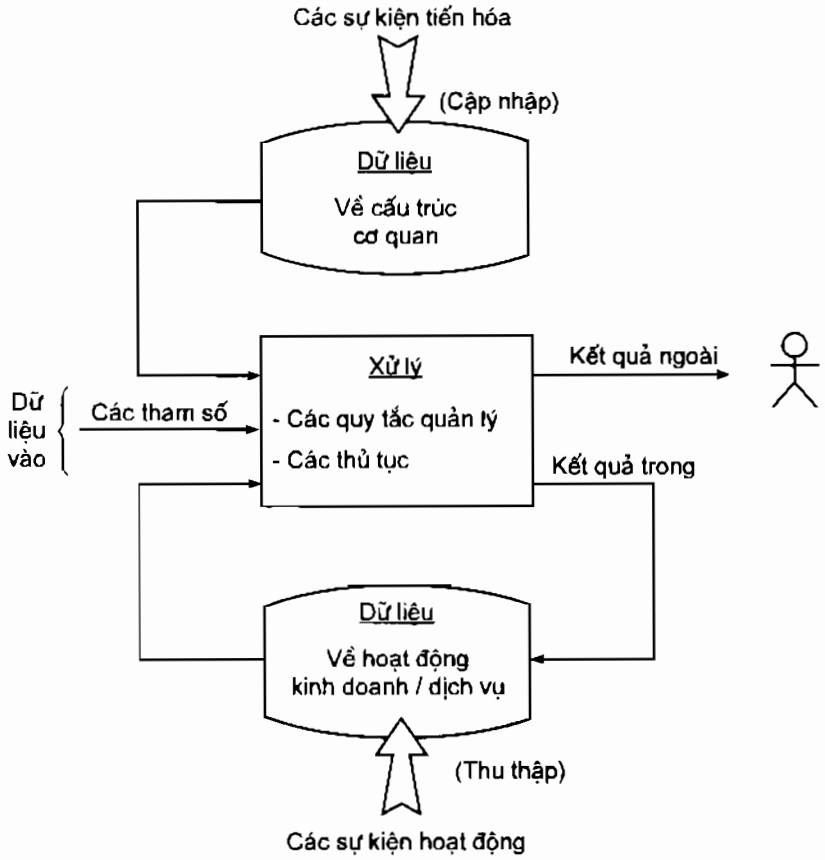
Mỗi xử lý thường là một sự áp dụng một *quy tắc quản lý* định sẵn và diễn ra theo một trật tự định sẵn (gọi là *thủ tục*). Các quy tắc quản lý và các thủ tục có thể được ấn định bởi hệ thống quyết định của doanh nghiệp, và như vậy chúng có thể bị điều chỉnh theo ý muốn (chẳng hạn các quy tắc tiêu thụ sản phẩm, phương pháp phân phối các trợ cấp, các quy định về khuyến mãi...), nhưng chúng có thể được ấn định từ bên ngoài doanh nghiệp, đặc biệt là bởi Nhà nước (ví dụ quy tắc tính thuế VAT, cách tính lương và bảo hiểm xã hội v.v...) và như vậy doanh nghiệp không được tùy tiện thay đổi.

Đầu vào của một xử lý có thể là các thông tin phản ánh cấu trúc doanh nghiệp và/ hoặc các thông tin phản ánh hoạt động của doanh nghiệp. Đầu ra có thể là:

+ Các kết quả chuyển trực tiếp cho các cá nhân hay tổ chức ngoài doanh nghiệp (chẳng hạn đơn đặt hàng, hóa đơn, thống kê bán hàng, báo cáo tài chính v.v...). Gọi đó là các *kết quả ngoài*;

+ Các kết quả được lưu giữ trở lại vào trong hệ thống, để sau này dùng làm đầu vào cho các xử lý khác (thường là các thông tin về tình trạng, về lịch sử hay lưu trữ). Gọi đó là các *kết quả trong*.

Tóm lại, có thể hình dung các thành phần cơ bản của hệ thống thông tin, với các mối liên quan về dữ liệu giữa chúng, như trong Hình I.3:



Hình I.3. Lược đồ về các thành phần của hệ thống thông tin trong doanh nghiệp.

§2. SỬ DỤNG MÁY TÍNH ĐỂ XỬ LÝ TỰ ĐỘNG CÁC THÔNG TIN

1. Các hệ thống tin học

Ở phần trên khi nói đến các hệ thống thông tin thì ta chưa quan tâm tới công cụ dùng để xử lý thông tin là gì (bằng tay hay bằng máy tính). Ở phần này ta mới xem xét đến vai trò của máy tính trong các hệ xử lý thông tin đó.

Theo từ điển Larousse Tin học, thì *Tin học* là tập hợp các ngành khoa học, kỹ thuật, kinh tế - xã hội vận dụng vào việc xử lý thông tin và sự tự động hóa nó". Nếu vậy, thì có thể định nghĩa *hệ thống tin học* là hệ thống có mục đích xử lý thông tin và có sự tham gia của máy tính.

Sự tham gia của máy tính trong một hệ thống tin học có thể ở nhiều mức độ khác nhau:

- mức thấp: máy tính chỉ được dùng để giải quyết một vài vụ việc đơn lẻ, như in một bảng biểu thống kê, lập một hóa đơn v.v...
- mức trung bình: máy tính cùng với con người cộng tác, phân công với nhau để thực hiện một quy trình quản lý phức tạp hay để giải quyết một vấn đề lớn.
- mức cao: máy tính đóng vai trò chủ chốt trong quá trình xử lý thông tin, con người không can thiệp vào quá trình này, và chỉ có nhiệm vụ cung cấp thông tin đầu vào cho máy tính và thu nhận các kết quả ra từ máy tính.

2. Các phương thức xử lý thông tin của máy tính

Việc xử lý thông tin bằng máy tính có thể thực hiện theo nhiều phương thức khác nhau:

a) Xử lý tương tác (interactive processing) và **xử lý giao dịch** (transactional processing)

Xử lý tương tác là xử lý thực hiện từng phần, xen kẽ giữa phần thực hiện bởi người và phần thực hiện bởi máy tính; hai bên trao đổi qua lại với nhau dưới hình thức đối thoại. Ở đây, con người không những đưa ra yêu cầu xử lý và cung cấp thông tin bổ sung khi cần, mà còn đưa ra các quyết định dẫn dắt quá trình để đi tới kết quả chung cục; máy tính trợ giúp cho quá trình đó. Trái lại trong xử lý giao dịch, xuất phát từ yêu cầu của người, máy tính thực hiện xử lý cho tới kết quả chung cục (một quá trình xử lý trọn vẹn như vậy của máy tính, không có ngưng ngắt, gọi là một *giao dịch* (transaction)). Có thể lưu ý rằng một quá trình xử lý tương tác I là một dãy các giao dịch T_i (i=1,n), trong đó kết quả ra của giao dịch T_j, cùng với các thông tin bổ sung và quyết định đưa ra từ phía con người sẽ là đầu vào cho giao dịch T_{j+1}.

b) Xử lý theo lô (batch processing) và **xử lý trực tuyến** (on-line processing)

Trong xử lý theo lô, mỗi khi thông tin đến (hay khi yêu cầu xử lý xuất hiện), thì chưa được đem xử lý ngay, mà được gom lại cho đủ một số lượng nhất định (một lô hay một mẻ) mới được đem xử lý một cách tập thể. Trong xử lý trực tuyến (hay còn gọi là xử lý trên dòng) thì thông tin đến được đem xử lý ngay lập tức, một cách cá thể và bất kể vào lúc nào. Xử lý theo lô thường áp

dụng cho các xử lý có tính định kỳ (hằng tuần, hằng tháng v.v...), cho các thống kê, các kết xuất, các báo cáo và cho việc in các chứng từ với khối lượng lớn (ví dụ in hóa đơn tiền điện ở một Chi cục quản lý điện). Xử lý trực tuyến thường áp dụng cho việc hiển thị, sửa chữa nội dung các tệp dữ liệu, cho việc phục vụ các giao dịch với khối lượng không nhiều, lại cần được thực hiện tại chỗ và cần có trả lời ngay (ví dụ bán vé máy bay).

c) Xử lý thời gian thực (real-time processing)

Là hành vi của một hệ thống phải thỏa mãn một số điều kiện ràng buộc rất ngặt nghèo về thời gian, chẳng hạn phải chịu hạn định đối với thời gian trả lời. Thông thường thì ở đây máy tính lệ thuộc vào một hệ thống ngoài (chẳng hạn một tên lửa, một lò nung,...), hệ thống này hoạt động theo một tiến độ riêng của nó và máy tính, với mục đích điều khiển sự hoạt động của hệ thống ngoài này, cần phải phản ứng một cách kịp thời đối với mọi biến động trạng thái của nó.

d) Xử lý phân tán (distributed processing)

Việc xử lý thực hiện trên mạng mà các nút là những đầu cuối nặng (nghĩa là các máy tính thực sự). Thông tin xuất hiện, có thể được xử lý một phần ở một đầu cuối, rồi được chuyển đi xử lý tiếp ở đầu cuối khác (hoặc ở bộ xử lý trung tâm). Ở đây, các cơ sở dữ liệu cũng có thể đặt rải rác ở các nút của mạng.

3. Một số loại hệ thống tin học thường gặp

Nếu xét tới nội dung của thông tin được xử lý và tính chất của môi trường của hệ thống, thì ta có thể phân biệt các loại hệ thống tin học khác nhau. Sau đây là một số loại hệ thống tin học điển hình:

a) Hệ thống thông tin quản lý (Management information systems)

Là hệ thống nhằm cung cấp các thông tin cần thiết cho sự quản lý, điều hành của một doanh nghiệp (hay nói rộng là của một tổ chức). Hạt nhân của hệ thống thông tin quản lý là một cơ sở dữ liệu chứa các thông tin phản ánh tình trạng hiện thời và hoạt động kinh doanh hiện thời của doanh nghiệp. Hệ thống thông tin thu thập các thông tin đến từ môi trường của doanh nghiệp, phối hợp với các thông tin có trong cơ sở dữ liệu để kết xuất các thông tin mà nhà quản lý cần, đồng thời thường xuyên cập nhật cơ sở dữ liệu để giữ cho các thông tin ở đó luôn phản ánh đúng thực trạng hiện thời của doanh nghiệp.

Các hệ thông tin quản lý thường được phân loại theo hai mức:

+ Mức thấp, hay còn gọi là mức tác nghiệp, hệ thống chỉ có nhiệm vụ in ra một số bảng biểu, chứng từ giao dịch theo khuôn mẫu của cách xử lý bằng tay truyền thống. Bấy giờ hệ thống thường được gọi là hệ xử lý dữ liệu (Data processing systems); đó thường là các hệ xử lý đơn hàng, hệ quản lý nhân sự, hệ quản lý thiết bị, hệ kế toán v.v...

+ Mức cao, hay còn gọi là mức điều hành, hệ thống phải đưa ra các thông tin có tính chất chiến lược và kế hoạch giúp cho người lãnh đạo doanh nghiệp đưa ra các quyết định đúng đắn trong công tác điều hành sự hoạt động của doanh nghiệp. Bấy giờ hệ thống thường được gọi là hệ hỗ trợ quyết định (Decision support systems). Đặc điểm của hệ hỗ trợ quyết định là, bên cạnh cơ sở dữ liệu, còn có thêm một cơ sở mô hình chứa các mô hình, các phương pháp mà khi được chọn lựa để vận dụng lên các dữ liệu sẽ cho các kết quả theo yêu cầu đa dạng của người dùng đặt ra khi chọn lựa các quyết định của mình.

b) Hệ thống tự động hóa sản xuất (Automatized manufactoring systems) hay còn gọi là **Hệ thống điều khiển quá trình** (Process control systems)

Đó là các hệ thống nhằm xử lý và điều khiển tự động các quá trình vận hành các thiết bị trong sản xuất, viễn thông, quân sự v.v... Các hệ thống này đều phải làm việc theo phương thức xử lý thời gian thực. Về mặt kiến trúc vật lý, thì bên cạnh các phần mềm, hệ thống này bao gồm nhiều loại thiết bị tin học đa dạng: từ các CPU phổ dụng thông thường, đến các máy tính chuyên dụng, các ôtômát lập trình được, như các PLC (Programmable logic controller), các đường truyền, các cảm biến, các kích hoạt, các bộ chuyển đổi A/N hay N/A...

c) Hệ thống nhúng thời gian thực (Embedded real-time systems)

Các hệ thống này được thực hiện trên các phần cứng đơn giản và nhúng trong một thiết bị nào đó, như mobiphone, ôtô, dụng cụ trong nhà v.v... Các hệ thống này thường được thực hiện bằng lập trình ở mức thấp, và cũng phải thực hiện xử lý theo thời gian thực. Trong các hệ này, thường thiếu vắng các ngoại vi thông dụng như màn hình, ổ đĩa cứng v.v...

d) Phần mềm hệ thống (System software)

Các hệ thống này thiết lập nên hạ tầng kỹ thuật của các hệ thống máy tính, phục vụ cho các phần mềm ứng dụng chạy trên đó. Đó có thể là hệ điều hành, chương trình dịch, hệ quản trị cơ sở dữ liệu, hệ giao diện người/ máy v.v... Chúng khai thác các dịch vụ tầng thấp của các phần cứng để đưa ra các giao diện khái lược, dễ sử dụng cho các chương trình ứng dụng.

e) Các hệ thống tự động hóa văn phòng (Automated Office systems)

Tự động hóa văn phòng là cách tiếp cận nhằm đưa máy tính vào hoạt động văn phòng, cho phép thâu tóm mọi việc tính toán, giao lưu, quản lý thông tin tất cả vào trong các cửa sổ trên màn hình máy tính, có ngay trên bàn làm việc của mỗi nhân viên. Một hệ thống tự động hóa văn phòng phải cung cấp được ít nhất một số trong các chức năng sau:

+ Thư tín điện tử: nhập các thông điệp văn bản gửi tới cá nhân hay nhóm;

+ Thư tín tiếng nói: nhập các thông điệp bằng lời nói, để người nhận có thể nghe lại sau đó;

+ Lịch biểu, bố trí thời gian, nhắc việc;

+ Các phương tiện tính toán đơn giản (loại máy tính bỏ túi) hay phức tạp (loại bảng tính);

+ Tích hợp điện thoại với tính toán: chuyển thành văn bản các cuộc gọi đến và bố trí sẵn các cuộc gọi đi, thông thường nhờ một CBX (computer-bazed branch exchange);

+ Quản lý tệp: các tệp cá nhân hay nhóm, các tệp ghi chép, các thư tín sao lưu...,

+ Hội thảo với tiếng nói hay dữ liệu: hội thảo nghe nhìn từ xa (nhiều người trên một hội thảo duy nhất), trao đổi dữ liệu (sự mở rộng của thư tín điện tử), các cuộc trò chuyện hỗn hợp dữ liệu và tiếng nói, nối ghép các màn hình với nhau;

+ Kết nối cửa sổ: chuyển giao nhanh chóng giữa các chức năng, không phải tốn chỗ;

+ Xử lý văn bản: soạn thảo, sửa chữa, mi trang... các tài liệu văn tự và đồ họa.

Ghi chú: Trong tập sách này, trừ Chương VI (Phân tích hệ thống về động thái) hướng tới các hệ thống thời gian thực, thì nội dung các chương còn lại đều chủ yếu hướng tới các hệ thống thông tin về quản lý. Bởi thế, từ nay khi nói tới hệ thống, thì ta có thể ngầm hiểu là hệ thống tin học, hay thậm chí là hệ thống thông tin trong quản lý cũng được.

§3. SỰ PHÁT TRIỂN HỆ THỐNG

Mọi hệ thống (tin học) đều phải trải qua sự khởi đầu, triển khai, xây dựng, khai thác, bảo dưỡng và kết thúc. Gọi quá trình đó là vòng đời (life cycle) hay nếu chỉ nhấn mạnh đến sự triển khai và xây dựng, thì gọi là sự phát triển của hệ thống (system development).

Để xem xét sự phát triển hệ thống, có hai khía cạnh phải đề cập đến:

- Sự tiếp nối các thời kỳ trong phát triển hệ thống, còn gọi là *chu trình phát triển* của hệ thống.
- Các phương tiện để nhận thức và diễn tả hệ thống, còn gọi là các *mô hình*.

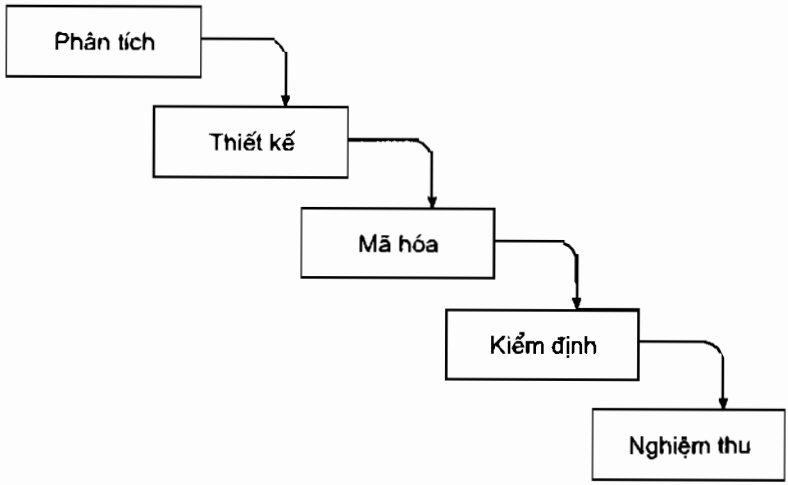
Dưới đây ta sẽ đề cập hai khía cạnh trên, trước khi xem xét đến các phương pháp phát triển hệ thống.

1. Chu trình phát triển

Có nhiều loại chu trình phát triển khác nhau. Sau đây là một số chu trình phát triển chính.

a) Chu trình thác nước

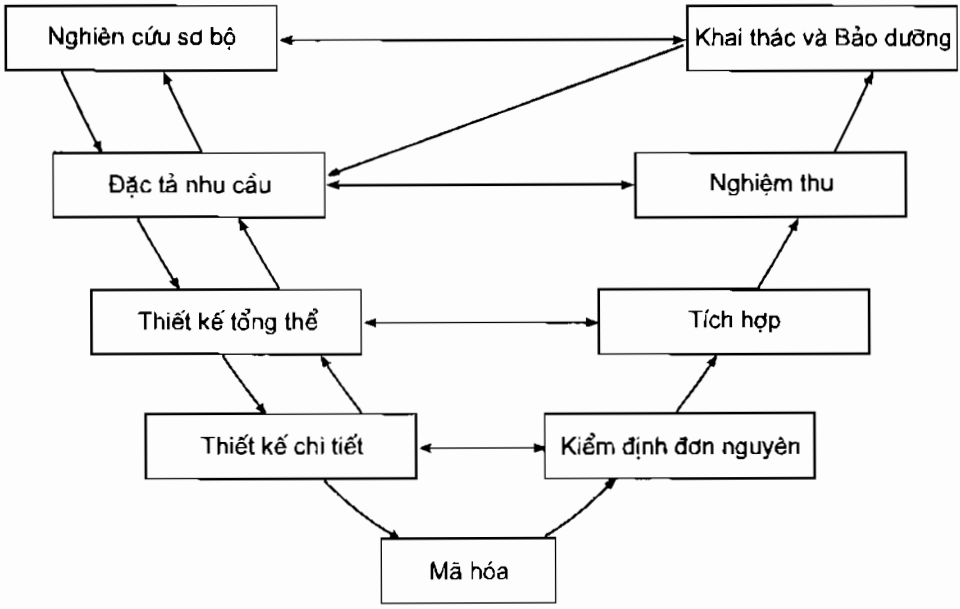
Đây là loại chu trình phát triển đầu tiên, được Royce đề xuất năm 1970, để mô tả sự phát triển của hệ thống. Đó là quá trình tiếp nối của năm giai đoạn: phân tích, thiết kế, mã hóa, kiểm định và nghiệm thu. Mỗi giai đoạn chỉ có thể bắt đầu khi giai đoạn trước đó đã hoàn tất (không được chờm lên nhau). Vì vậy chu trình phát triển này còn được gọi là chu trình tuyến tính (Hình I.4).



Hình I.4. Chu trình thác nước

Nhược điểm chính của chu trình phát triển thác nước là ở chỗ không có sự quay lui. Nhưng sự quay lui lại là một nhu cầu rất tự nhiên, vì nhiều khi có vào giai đoạn sau thì ta mới phát hiện được những thiếu sót bắt nguồn từ giai đoạn trước và cần quay lui để chỉnh sửa lại.

Chính vì vậy mà cũng đã có nhiều phương án cải tiến của chu trình thác nước, cho phép sự quay lui. Chẳng hạn chu trình phát triển hình chữ V, được AFCIQ (Association Française pour le Contrôle Industriel de la Qualité) đề nghị bao gồm cả các bước quay lui, và ngoài ra đặt tương ứng các giai đoạn kiểm nghiệm (khai thác - bảo dưỡng, nghiệm thu, tích hợp, kiểm định đơn nguyên) với các giai đoạn phân tích và thiết kế (nghiên cứu sơ bộ, đặc tả nhu cầu, thiết kế tổng thể, thiết kế chi tiết). Giai đoạn mã hóa là ở đáy của chữ V. Nhánh bên trái chữ V gồm các giai đoạn phân tích và thiết kế. Nhánh bên phải gồm các giai đoạn kiểm nghiệm. Khi một sai sót được phát hiện trong một giai đoạn kiểm nghiệm, thì giai đoạn phân tích hay thiết kế tương ứng được xem lại và chu trình bắt đầu lại từ đó.



Hình I.5. Chu trình phát triển hình V

b) Chu trình tăng trưởng

Chu trình tăng trưởng, do D. R. Graham đề xuất năm 1989, dựa trên các bước tăng trưởng dần dần, cho phép hoàn thành hệ thống từng mảnh một. Mỗi bước tăng trưởng thực hiện một tiến trình tuyến tính để triển khai một phần có thể chuyển giao được của cả hệ thống. Quy trình này lặp lại nhiều lần cho tới

- Chính xác hóa các nhu cầu phải đáp ứng: Thường thì các nhu cầu của người dùng không được phát biểu một cách rành mạch. Nhà tin học có thể hiểu chúng lệch lạc theo chủ quan của mình. Một nguyên mẫu sẽ phô diễn một cách cụ thể để người dùng thấy rằng nó có đáp ứng trúng nhu cầu của mình không.
- Phát hiện các hành vi lệch lạc, các sai lỗi: Trong sơ đồ thiết kế, có thể có những điểm tế nhị, người thiết kế không lường hết được mọi tình huống. Thực hiện nguyên mẫu ta có thể phát hiện các hành vi lệch lạc, các sai lỗi của hệ thống.
- Đánh giá được hiệu năng của hệ thống: Đương nhiên hiệu năng của hệ thống liên quan chặt chẽ tới sự thích ứng của ngôn ngữ lập trình và máy tính được sử dụng. Rất nhiều khi ngôn ngữ và máy tính được dùng cho cài đặt hệ thống sau này lại chưa có ngay, và ta phải làm nguyên mẫu trên ngôn ngữ và máy tính khác, để tranh thủ thời gian. Ngay cả lúc đó nguyên mẫu cũng phản ánh hiệu năng (tương đối) của chương trình, và thông qua nguyên mẫu ta cũng có thể phát hiện các nguyên nhân của sự chậm chạp từ bên trong chương trình. Thường thì nguyên nhân cơ bản nằm ở khâu xử lý hàng loạt trong chương trình. Còn những khâu như tương tác người/ máy chẳng hạn, có thể làm mất thời gian nhưng cái chính là do phải chờ đợi sự trả lời của con người, thì vẫn có thể được giữ y nguyên.

Kỹ thuật làm nguyên mẫu ngày nay thực hiện được là nhờ có các ngôn ngữ lập trình phi thủ tục, còn gọi là ngôn ngữ lập trình thế hệ bốn.

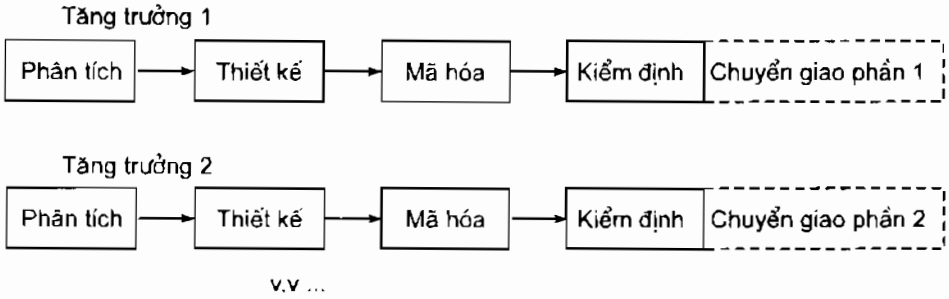
Với việc làm nguyên mẫu thì tiến trình triển khai dự án sẽ có nhiều khác biệt so với tiến trình tuyến tính. Chu trình là ngắn hơn nhiều, song được lặp đi lặp lại nhiều lần. Theo Jenkins, Milton và Naumann (Đại học Indiana City), thì có bốn giai đoạn cho một vòng lặp (Hình I.7):

Giai đoạn 1: Với vòng lặp đầu tiên thì giai đoạn này nhằm phát hiện các nhu cầu cơ bản và rõ rệt nhất, thông qua các phương pháp thông thường như phỏng vấn, xem tài liệu v.v... Không cần phải tìm vét cạn, mà cố chuyển nhanh sang giai đoạn sau. Từ vòng lặp thứ hai trở đi, thì giai đoạn này nhằm xác định các mục tiêu của vòng lặp hiện tại, các phương án có thể để đạt các mục tiêu đó và các ràng buộc từ kết quả vòng lặp trước.

Giai đoạn 2: Đánh giá các phương án có thể, bằng cách phát hiện các nguy cơ tiềm ẩn và cách giải quyết chúng.

Giai đoạn 3: Thiết kế và thành lập một nguyên mẫu chạy được. Ở đây cũng vậy, cần ưu tiên sự nhanh chóng bằng cách tập trung vào cái cốt yếu, hoãn lại sau sự tinh chỉnh các màn hình hay các biểu mẫu in ra.

khi có một phương án hoàn chỉnh của cả hệ thống. Rõ ràng là cách làm này chỉ thích hợp với các hệ thống có thể chia cắt và chuyển giao theo từng mảnh.



Hình I.6. Chu trình phát triển tăng trưởng

c) Chu trình xoắn ốc

Chu trình xoắn ốc hay chu trình lặp là do Bochm đề xuất năm 1988, với các đặc điểm sau:

- Tiến trình lặp đi lặp lại một dãy các giai đoạn nhất định;
- Qua mỗi vòng lặp, tạo ra một nguyên mẫu hoàn thiện dần;
- Nhấn mạnh sự khắc phục các nguy cơ (một nguy cơ bắt nguồn từ các sai sót trong sự đặc tả các nhu cầu).

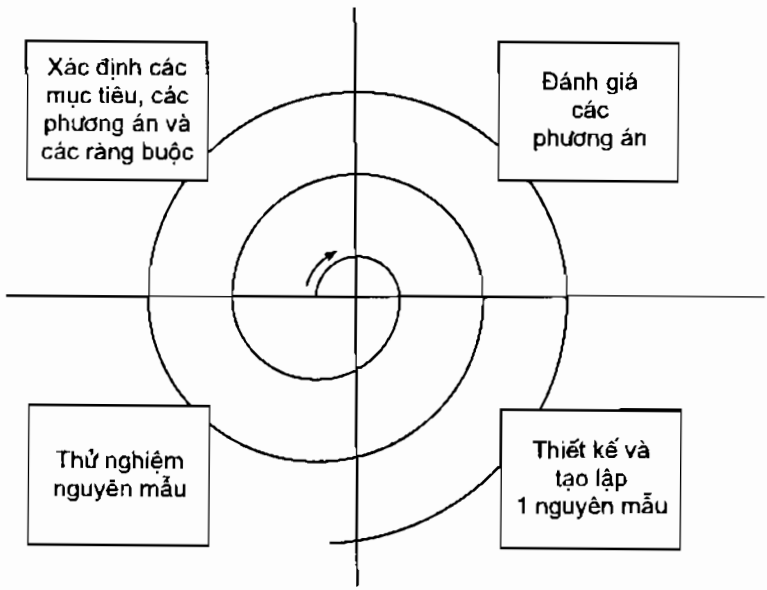
Theo từ điển Petit Larousse, thì nguyên mẫu (prototype) là phiên bản đầu tiên, được tạo lập một cách công nghiệp, của một tổ hợp (thiết bị, máy...), nhằm để thử nghiệm về chất lượng trước khi sản xuất hàng loạt.

Trong tin học, thì nghĩa của thuật ngữ này có thay đổi chút ít. Một phần mềm nguyên mẫu là một hệ thống:

- có khả năng làm việc được trên các dữ liệu thực, nghĩa là nó đã vượt quá giai đoạn dự án trên giấy, và như thế nó có thể được đánh giá bởi người thiết kế và/ hoặc các người dùng;
- có thể được phát triển thêm để tiến tới hệ thống kết cục, hoặc có thể dùng làm cơ sở cho việc thực hiện nó;
- được tạo lập nhanh và ít tốn kém;
- được dùng để kiểm chứng các giả định về các nhu cầu phải đáp ứng, về các lược đồ thiết kế hoặc về lôgic của các chương trình.

Như vậy, việc tạo ra các nguyên mẫu nhanh là có ích trên nhiều phương diện:

Giai đoạn 4: Thử nghiệm nguyên mẫu. Trước hết giới thiệu nó cho một số nhỏ các người dùng tuyển chọn, để thu thập ở họ các phê phán, các góp ý. Tùy theo mức độ quan trọng, một số điều chỉnh được thực hiện tại chỗ, một số hoãn lại đến vòng lặp tiếp sau. Nếu hệ thống tỏ ra khá khác biệt với sự mong muốn của các người dùng, có thể gạt bỏ nó ngay để làm lại từ đầu. Nếu hệ thống tỏ ra có triển vọng làm điểm xuất phát được, ta sẽ chỉ cho các người dùng thấy là nó sẽ được điều chỉnh ra sao, rồi để người dùng thử vận hành trong một thời gian và ghi nhận tiếp các yêu cầu điều chỉnh.



Hình I.7. Chu trình xoắn ốc

Các vòng lặp được tiếp tục cho đến khi xét thấy nguyên mẫu là tốt để có thể chuyển sang sản xuất thực sự được.

Có người e rằng cách làm vòng vo sẽ làm kéo dài thời gian. Không phải vậy, một nghiên cứu nghiêm túc của Boehm, Gray và Seewald cho thấy thời gian có thể rút xuống còn khoảng 45% so với cách làm cũ.

Mặt khác sự thành công của tiến trình lặp có thể dẫn tới một vài hậu quả tiêu cực cần dè chừng. Người dùng có thể thỏa mãn với vài phương án đầu và muốn dừng ngay, mặc dù không phải là không còn những việc đáng làm. Người xây dựng hệ thống có thể say sưa với thành công nhanh chóng, mà sinh ra kém

nghiêm túc trong nghề nghiệp, không giữ đúng tiến độ công việc theo kế hoạch, bỏ quên việc kiểm chứng hay một số việc khác, vốn ít có ảnh hưởng trực tiếp tới kết quả trước mắt. Việc làm tư liệu, vốn rất cần thiết cho sự hoạt động và bảo trì hệ thống sau này, cũng dễ bị bỏ qua hay xem nhẹ. Bởi vì trong tiến trình tuyến tính, thì hoàn thành tư liệu cho mỗi giai đoạn là điều kiện tiên quyết để chuyển sang giai đoạn tiếp; còn trong tiến trình lặp không thể làm tư liệu sau mỗi vòng lặp được, vì mọi thứ đều còn dở dang, mà phải làm tư liệu vào thời điểm cuối, gây ra không ít e ngại.

d) Chu trình lắp ráp các thành phần

Chu trình lắp ráp các thành phần dựa trên việc sử dụng lại các thành phần phần mềm. Việc tạo lập hệ thống được thực hiện bằng cách lắp ráp các thành phần có sẵn. Như vậy điều quan trọng là cần phải xác định và thu gom các thành phần có khả năng sử dụng lại càng sớm càng tốt. Các thành phần khác, tham gia vào hệ thống có thể là được điều chỉnh, thích ứng từ các thành phần có sẵn hoặc được xây dựng từ số không.

Theo Hooper, Chester và Kang thì tiến trình gồm 6 giai đoạn:

- Nhận thức bài toán: Tìm hiểu vấn đề được đặt ra và khả năng sử dụng lại.
- Hình thành giải pháp: Đề xuất một số giải pháp trên hướng sử dụng các thành phần có sẵn.
- Tìm kiếm các thành phần: Tuyển chọn các thành phần thích hợp.
- Điều chỉnh và thích ứng các thành phần: Điều chỉnh các thành phần làm cho nó thực sự thích ứng với giải pháp.
- Tích hợp các thành phần: Lắp ráp các thành phần thành giải pháp.
- Đánh giá: Đánh giá kết quả thực hiện, đồng thời xác định các thành phần mới có thể lưu để sử dụng lại sau này.

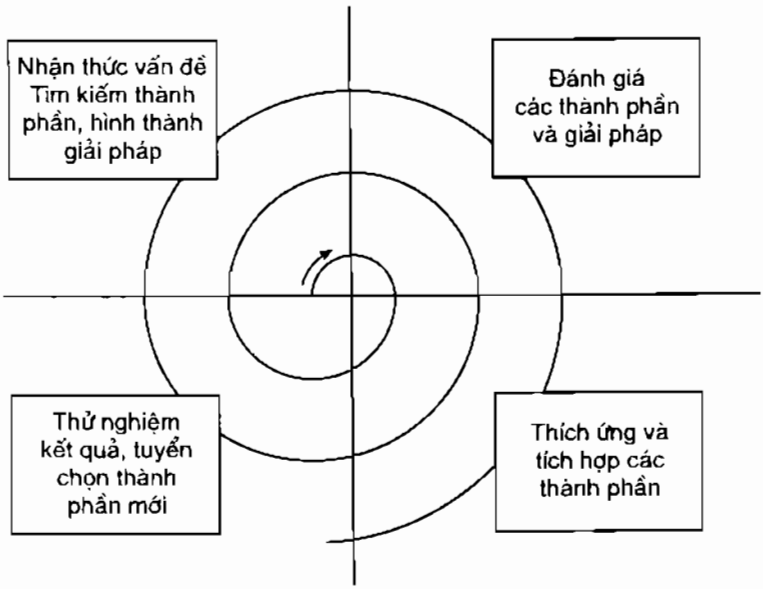
Cũng có thể tổ chức tiến trình nói trên thành chu trình lặp, với bốn giai đoạn tương ứng với bốn giai đoạn nói ở mục trên, như sau (Hình I.8), mỗi lần lặp tạo nên một mảnh của hệ thống:

+ Giai đoạn 1: Nhận thức vấn đề và hình thành giải pháp trên cơ sở sử dụng lại. Các kết quả của giai đoạn này là một tập hợp các thành phần có khả năng áp dụng cho hệ thống đang tạo lập cùng với các giải pháp dựa trên chức năng của các thành phần đó.

+ Giai đoạn 2: Đánh giá các giải pháp và tuyển chọn các thành phần thích hợp. Kết quả của giai đoạn này là các thành phần được sử dụng lại cùng các yêu cầu về điều chỉnh và thích ứng đối với chúng.

+ Giai đoạn 3: Thích ứng và tích hợp các thành phần. Kết quả của giai đoạn này là tạo được một phần nào đó (một hệ con) của hệ thống toàn thể trên cơ sở lắp ráp các thành phần.

+ Giai đoạn 4: Thử nghiệm và đánh giá kết quả. Qua đó tuyển một số thành phần có thể đưa vào kho thành phần sử dụng lại.



Hình I.8. Chu trình lặp của việc sử dụng lại

2. Mô hình hóa hệ thống

Các bước phát triển hệ thống như là tìm hiểu nhu cầu, phân tích hệ thống và thiết kế hệ thống tuy có khác nhau về nhiệm vụ và mục tiêu, song chúng có đặc điểm chung:

- đều phải đối đầu với sự phức tạp;
- đều là những quá trình *nhận thức* và *diễn tả* sự phức tạp thông qua các mô hình.

Nói cách khác đó đều là những quá trình mô hình hóa.

a) Nguyên lý chế ngự sự phức tạp

Để tìm hiểu một thế giới vô cùng phức tạp, mọi khoa học thực nghiệm đều vận dụng chung một nguyên lý cơ bản, đó là sự trừu tượng hóa (abstraction).

> *Trừu tượng hóa* (hay còn gọi là *trừu xuất*) là một nguyên lý của sự nhận thức, đòi hỏi phải bỏ qua các sắc thái (của một chủ đề) không liên quan tới chủ định hiện thời, để tập trung hoàn toàn vào các sắc thái liên quan tới chủ định đó (từ điển Oxford).

Nói cách khác, trước một bài toán (hay một chủ đề), ta tạm quên hay lờ đi các chi tiết có tác dụng rất ít hoặc không có tác dụng gì đối với lời giải bài toán, nhờ đó hình thành được một sự diễn tả đơn giản hóa và dễ hiểu, cho phép chúng ta giải quyết được bài toán, vẫn theo đúng bản chất của nó.

b) Mô hình

> *Mô hình* (model) là một dạng trừu tượng hóa của một hệ thống thực.

Nói rõ hơn, thì mô hình là một hình ảnh (một biểu diễn) của một hệ thống thực, được diễn tả:

- ở một mức độ trừu tượng hóa nào đó,
- theo một quan điểm (hay một góc nhìn) nào đó,
- bởi một hình thức hiểu được (văn bản, phương trình, bảng, đồ thị, v.v...) nào đó.

Việc dùng mô hình để nhận thức và diễn tả một hệ thống gọi là *mô hình hóa*.

Ngày nay các phương pháp phân tích và thiết kế hệ thống đều có xu hướng là sử dụng các mô hình dạng *biểu đồ* (diagrams). Các biểu đồ đều là những đồ thị, trong đó các nút và các cung được vẽ theo các dạng riêng biệt và mang các ý nghĩa riêng biệt, tùy theo yêu cầu diễn tả của chúng.

c) Mục đích và chất lượng của mô hình hóa

Có ba mục đích:

- Mô hình hoá để hiểu: Nói cho cùng thì hiểu tức là hình thành được một hình ảnh xác thực và giản lược (ở trong đầu hay trên giấy) về đối tượng được tìm hiểu. Vậy "hiểu tức là mô hình hóa" (Jean Piaget). Không thể nói rằng hiểu mà chưa có mô hình. Ngược lại, biết vận dụng các loại mô

hình, ta sẽ nhận thức vấn đề dễ dàng và nhanh chóng hơn.

- Mô hình hóa để trao đổi: Vì tính hiểu được của mô hình mà nó trở thành một thứ ngôn ngữ chung cho phép trao đổi giữa những người cùng quan tâm tới một vấn đề hay một hệ thống chung.
- Mô hình hóa để hoàn chỉnh: Nhờ sự minh bạch của mô hình mà ta dễ nhận thấy hệ thống đã phù hợp với nhu cầu chưa, có chặt chẽ, có đầy đủ không, nhờ đó mà có thể hoàn thiện thêm. Hơn nữa, mô hình còn giúp ta kiểm định, mô phỏng, thực hiện.

Từ ba mục đích trên, ta suy ra một mô hình tốt phải có các đặc điểm sau:

- dễ đọc,
- dễ hiểu,
- dễ trao đổi,
- xác thực,
- chặt chẽ,
- đầy đủ,
- dễ thực hiện.

d) Hai mức độ mô hình hóa hệ thống

Mọi mô hình đều phản ánh hệ thống theo một mức độ trừu tượng hóa nào đó. Thường thì người ta phân biệt hai mức độ chính:

- *Mức lôgic*: tập trung mô tả bản chất của hệ thống và mục đích hoạt động của hệ thống, mà bỏ qua các yếu tố về tổ chức thực hiện, về biện pháp cài đặt. Nói cách khác, mô hình lôgic trả lời các câu hỏi "Là gì?" (What?) - như là chức năng gì? thông tin gì? ứng xử gì? - mà bỏ qua các câu hỏi "Như thế nào?" (How?)
- *Mức vật lý:* Trả lời các câu hỏi "Như thế nào", quan tâm tới các mặt như: phương pháp, biện pháp, công cụ, tác nhân, địa điểm, thời gian, hiệu năng v.v...

Chính vì có sự phân biệt hai mức độ mô hình hóa (lôgic và vật lý) như trên mà mọi quá trình phát triển hệ thống, dù là theo chu trình sống nào, cũng đều phải bao gồm hai giai đoạn trung tâm và phân biệt, là phân tích và thiết kế.

- Giai đoạn *phân tích hệ thống* có mục đích đi sâu vào bản chất và chi tiết của hệ thống. Nó giải đáp câu hỏi "Là gì?" mà bỏ qua câu hỏi "Như thế

nào?". Vậy phân tích quan tâm vấn đề mà không quan tâm giải pháp. Vấn đề nói đây thường là: chức năng, dữ liệu và động thái của hệ thống.

- Giai đoạn *thiết kế hệ thống* lại có mục đích là chọn lựa các giải pháp cài đặt, nhằm hiện thực hóa các kết quả phân tích (cho nên thiết kế phải đi sau phân tích và đi trước lập trình). Nó giải đáp câu hỏi "Như thế nào?" Thiết kế luôn luôn phải tìm sự dung hòa giữa tính hợp lý của các kết quả phân tích với các yêu cầu của thực tiễn, như là các ràng buộc, các hạn chế, các ưu tiên, sự nhanh chóng và sự tiện dùng. Kiến trúc vật lý của hệ thống sẽ được làm rõ trong giai đoạn này.

d) Bốn trục mô tả của mô hình hóa

Thực ra mô hình lôgic hay mô hình vật lý của hệ thống bao gồm nhiều mô hình con. Mỗi mô hình con mô tả hệ thống về một phương diện, hay theo một góc nhìn (quan điểm) nào đó.

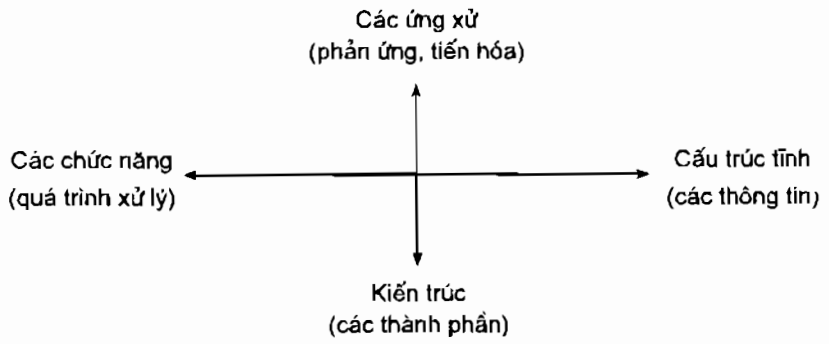
Các phương pháp kinh điển thường phân biệt bốn phương diện mô tả hệ thống (thường gọi là bốn trục mô hình hóa):

- Mô tả các *chức năng* mà hệ thống phải thực hiện;
- Mô tả các *đặc điểm tĩnh* của hệ thống: các thông tin lưu giữ; các yếu tố tạo nên cấu trúc (các quan hệ);
- Mô tả cách *ứng xử* (hay *động thái*) của hệ thống, gồm :

  + các phản ứng (tức thời)

  + các tiến hóa (trong thời gian dài);

- Mô tả các *thành phần* (thuộc phần mềm và phần cứng), từ đó xây dựng kiến trúc của hệ thống.



Hình I.10 Bốn trục diễn tả trong mô hình hóa.

3. Các phương pháp mô hình hóa hệ thống

Ngày nay tồn tại rất nhiều phương pháp mô hình hóa hệ thống (cũng còn được gọi là các phương pháp phân tích và thiết kế hệ thống). Người phát triển hệ thống, trước khi bắt tay vào việc, phải chọn lựa một phương pháp thích hợp với mình và với hệ thống cần xây dựng.

a) Ba thành phần cơ bản của một phương pháp

Một phương pháp là một sự tổng hợp của ba thành phần:

- Tập hợp các khái niệm và mô hình: Mỗi phương pháp đều phải dựa trên một số không nhiều các khái niệm cơ bản, và sử dụng một số dạng mô hình nhất định, kèm với các kỹ thuật để triển khai hay biến đổi các mô hình đó. Chẳng hạn phương pháp SA (Structured Analysis) dựa trên các khái niệm "đối tác", "chức năng", "luồng dữ liệu", "kho dữ liệu"; các mô hình chính mà nó dùng là biểu đồ luồng dữ liệu và từ điển dữ liệu; nó đưa ra các kỹ thuật biến đổi từ biểu đồ luồng dữ liệu vật lý sang biểu đồ luồng dữ liệu lôgic, từ biểu đồ luồng dữ liệu hệ thống cũ sang biểu đồ luồng dữ liệu hệ thống mới.

- Một tiến độ triển khai: bao gồm các bước đi lần lượt, các hoạt động cần làm, các sản phẩm qua từng giai đoạn (như tư liệu, mô hình....), cách điều hành đối với tiến độ đó và cách đánh giá chất lượng các kết quả thu được. Chẳng hạn phương pháp hướng đối tượng OOA/D của Coad và Yourdon triển khai giai đoạn phân tích theo năm tầng lần lượt là: lớp và đối tượng, cấu trúc, chủ đề, thuộc tính, dịch vụ; tiếp đó triển khai giai đoạn thiết kế theo bốn thành phần lần lượt là: giao diện người máy, lĩnh vực bài toán, quản lý các nhiệm vụ, quản lý các dữ liệu.

- Các công cụ trợ giúp: Đó là các phần mềm hỗ trợ cho quá trình mô hình hóa với các khả năng sau:

  + sản sinh các mô hình và biểu đồ;

  + biến đổi và điều chỉnh nhanh các mô hình và biểu đồ;

  + kiểm tra cú pháp, sự chặt chẽ, sự đầy đủ;

  + kiểm thử vá đánh giá;

  + mô phỏng và thực hiện mô hình.

b) Một số phương pháp mô hình hóa có tiếng

Người ta thường phân loại các phương pháp mô hình hóa theo hai trào lưu chính: mô hình hóa hướng chức năng (lấy chức năng làm trục mô hình hóa

chính) và mô hình hóa hướng đối tượng (lấy đối tượng làm đơn vị mô hình hóa). Tuy nhiên ta có thể phân loại chi tiết hơn và liệt kê các phương pháp (có tiếng) như sau:

- Các phương pháp "hệ thống":
  + MERISE (H. Tardieu, A. Rochfeld 1976);
- Các phương pháp chức năng hay có cấu trúc:
  + SA (De Marco , 1978);
  + SADT (Douglas T. Ross, 1977);
  + SA-RT (Ward-Mellor, 1985; Hatley-Pirbhai, 1987);
- Phương pháp theo sự kiện:
  + State Charts (D. Harel, 1987);
  + Phương pháp tích hợp (O. Foucaut, O. Thiéry, 1996);
- Các phương pháp hướng dữ liệu:
  + LCP, LCS (J.D. Warnier, 1969-70);
  + E/A (H. Tardieu, P. Chen, 1976);
- Các phương pháp hướng đối tượng:
  + OOA/RD (Shlaer - Mellor, 1991-92);
  + OOAD (G. Booch, 1992-93);
  + OMT (J. Rumbaugh, 1992);
  + OOA/OOD (P. Coad, E. Yourdon, 1991);
  + OOSE (I. Jacobson, 1992);
  + Fusion (D. Coleman, 1994 - từ Hewlett - Packard);
  + SOART (P. Lavret, 1994);
  + UML + RUP + Rational Rose (G. Booch, J. Rumbaugh, I. Jacobson, 1997).

c) Những thách thức đối với các phương pháp mô hình hóa

Có ba thách thức chính:

- Sự phức tạp của lĩnh vực bài toán và của trách nhiệm của hệ thống: Lĩnh vực của bài toán thường bao gồm những nghiệp vụ phức tạp và xa lạ đối với người phát triển hệ thống (ví như kế toán, ngân hàng, quản lý bay,

điều khiển ra đa...), mặt khác trách nhiệm của hệ thống cũng rất đa dạng đòi hỏi phương pháp mô hình hóa được dùng phải giúp cho người phát triển hệ thống nhanh chóng và dễ dàng chế ngự được sự phức tạp, nắm bắt được các tình huống và vấn đề.

- Yêu cầu trao đổi giữa người với người: Nói đến công nghệ phần mềm, người ta thường nghĩ tới dữ liệu, xử lý, thuật toán, máy tính v.v... mà quên mất yếu tố quan trọng là con người và sự giao lưu của người. Bài toán là do con người đặt ra và phải được giải quyết bởi con người. Phương pháp mô hình hóa tốt phải là cầu nối tốt cho sự trao đổi giữa người phát triển hệ thống với người dùng và với các đồng nghiệp của mình.

- Đối đầu với sự thay đổi liên tục: Sự thay đổi các nhu cầu dẫn tới sự thay đổi nền móng của hệ thống là thường xuyên xảy ra. Phải xem đó là chuyện bình thường và phải chấp nhận, bởi vì các thay đổi đó bắt nguồn từ những áp lực không thể trốn tránh: các khách hàng, sự cạnh tranh, các người làm luật, sự phát triển của kỹ thuật v.v... Phương pháp mô hình hóa tốt phải lấy các yếu tố ổn định làm nền tảng cho các mô hình của mình, còn các yếu tố dễ thay đổi (thường gọi là các yếu tố dễ bay biến - volatile) thì được phản ánh trong mô hình với sự khoanh bọc vào các phạm vi nhỏ hẹp, sao cho nếu xảy ra thay đổi và điều chỉnh thì ít có ảnh hưởng đến đại cục, không gây ra sự đổ vỡ hệ thống.

d) Các phương pháp mô hình hóa được chọn để giới thiệu trong tập sách

Tập sách này được xây dựng từ một giáo trình "Nhập môn phân tích và thiết kế hệ thống thông tin", dành cho các sinh viên ở bậc đại học. Nó trình bày các phương pháp kinh điển (hướng chức năng) thay vì các phương pháp hiện đại (hướng đối tượng). Các lý do chính là:

- Các phương pháp kinh điển là dễ hiểu, dễ thực hành, đồng thời vẫn chuyển tải được đến người đọc nhiều tư tưởng cơ bản của công việc phân tích và thiết kế, cho nên chúng là thích hợp đối với người mới "nhập môn".

- Các phương pháp kinh điển có bề dày ứng dụng trên nhiều dự án, nhiều nước, và đến nay vẫn còn được tiếp tục vận dụng, đặc biệt là ở nước ta. Một số phần mềm trợ giúp (CASE) nổi tiếng như Designer 2000 của ORACLE vẫn tiếp tục đi theo hướng kinh điển. Vậy học các phương pháp này không chỉ để biết, mà còn để dùng thực sự.

Vì không có một phương pháp kinh điển xuyên suốt mọi khâu trong quá trình mô hình hóa hệ thống (nếu có, lại là phương pháp quá cầu kỳ và ôm đồm, như MERISE), cho nên ở đây ta kết nối nhiều phương pháp lại. Tuy nhiên, các phương pháp được chọn lựa này đều có chung một tư tưởng chỉ đạo: đó là tư tưởng phân tích và thiết kế trên xuống, còn gọi là phân tích và thiết kế có cấu trúc.

Việc phân tích hệ thống về chức năng (Chương III) sẽ được trình bày theo phương pháp SA.

Việc phân tích hệ thống về dữ liệu (Chương IV và Chương V) sẽ được trình bày theo phương pháp E/A (bổ sung bằng mô hình quan hệ).

Việc phân tích hệ thống về động thái (Chương VI) sẽ được trình bày theo phương pháp SART.

Việc thiết kế hệ thống (Chương VII) sẽ được trình bày theo phương pháp SD.

Trật tự trình bày các chương (tìm hiểu nhu cầu, phân tích chức năng, phân tích dữ liệu, phân tích động thái, thiết kế, lập trình và kiểm định) dường như là sự thể hiện một chu trình phát triển tuyến tính. Thực ra, đó là sự gom nhóm theo chủ đề để trình bày cho tiện, chứ không phải là sự áp đặt một chu trình phát triển nào cả.

Cuối cùng thì cũng cần phải nói thêm là: Ngày nay người làm tin học có tay nghề không thể trốn tránh lập trình hướng đối tượng, vậy cũng không thể không biết về mô hình hóa hướng đối tượng. Tuy nhiên, độc giả có thể tìm hiểu chủ đề này sau, cũng như tác giả sẽ còn nuôi ý định dành chủ đề này cho một tập sách sau.

Chương II

KHẢO SÁT HIỆN TRẠNG VÀ TÌM HIỂU CÁC NHU CẦU

Đây là bước mở đầu của quá trình phát triển hệ thống, còn gọi là bước phát biểu bài toán, đặt vấn đề, hay nghiên cứu sơ bộ.

"Nhu cầu là mẹ của mọi sự sáng tạo". Cho nên, để sáng tạo ra một hệ thống mới, người phát triển hệ thống trước hết phải làm quen và thâm nhập vào chuyên môn nghiệp vụ mà hệ thống đó phải đáp ứng, tìm hiểu các nhu cầu đặt ra đối với hệ thống đó, tập hợp các thông tin cần thiết cho phép giải đáp một số câu hỏi cơ bản, như:

- Môi trường, hoàn cảnh, các ràng buộc và hạn chế đối với hệ thống đó như thế nào?
- Chức năng, nhiệm vụ và mục tiêu cần đạt được của hệ thống đó là gì, tức là người dùng thực sự muốn gì ở hệ thống?
- Có thể hình dung sơ bộ một giải pháp có thể đáp ứng được các yêu cầu đặt ra như thế nào?

Tóm lại, thì một dự án triển khai một hệ thống mới là có thực sự cần thiết và khả thi không? Kế hoạch và tiến độ triển khai dự án đó ra sao?

Sau đây ta sẽ lần lượt xét những công việc chính phải làm trong giai đoạn bước đầu này, một giai đoạn được gọi là nghiên cứu sơ bộ, song chẳng phải là tốn ít công sức cho nó.

§1 KHẢO SÁT VÀ ĐÁNH GIÁ HIỆN TRẠNG

1. Đại cương

a) Mục đích khảo sát hiện trạng

Thông thường thì một hệ thống mới được xây dựng là nhằm để thay thế cho một hệ thống cũ đã bộc lộ nhiều điều bất cập. Chính vì vậy mà việc tìm hiểu nhu cầu đối với hệ thống mới thường bắt đầu từ việc khảo sát và đánh giá hệ thống cũ đó. Vì rằng hệ thống này đang tồn tại, đang hoạt động nên ta gọi đó là hiện trạng. Việc khảo sát hiện trạng là nhằm để:

- Tiếp cận với nghiệp vụ chuyên môn, môi trường hoạt động của hệ thống.
- Tìm hiểu các chức năng, nhiệm vụ và cung cách hoạt động của hệ thống.
- Chỉ ra các chỗ hợp lý của hệ thống, cần được kế thừa và các chỗ bất hợp lý của hệ thống, cần được nghiên cứu khắc phục.

b) Nội dung khảo sát và đánh giá hiện trạng

Việc khảo sát được thực hiện theo các nội dung sau:

- Tìm hiểu môi trường xã hội, kinh tế và kỹ thuật của hệ thống; nghiên cứu cơ cấu tổ chức của cơ quan chủ quản hệ thống đó.
- Nghiên cứu các chức trách, nhiệm vụ, các trung tâm ra quyết định và điều hành, sự phân cấp các quyền hạn.
- Thu thập và nghiên cứu các hồ sơ, sổ sách, các tệp cùng với các phương thức xử lý các thông tin trong đó.
- Thu thập và mô tả các quy tắc quản lý, tức là các quy định, các công thức do nhà nước hoặc cơ quan đưa ra làm căn cứ cho các quá trình xử lý thông tin.
- Thu thập các chứng từ giao dịch và mô tả các chu trình lưu chuuyển và xử lý các thông tin và tài liệu giao dịch.
- Thống kê các phương tiện và tài nguyên đã và có thể sử dụng.
- Thu thập các đòi hỏi về thông tin, các ý kiến phê phán, phàn nàn về hiện trạng, các dự đoán, nguyện vọng và kế hoạch cho tương lai.
- Đánh giá, phê phán hiện trạng; đề xuất hướng giải quyết.
- Lập hồ sơ tổng hợp về hiện trạng.

c) Các yêu cầu đối với một cuộc điều tra

Việc khảo sát, điều tra phải đạt các yêu cầu như sau:

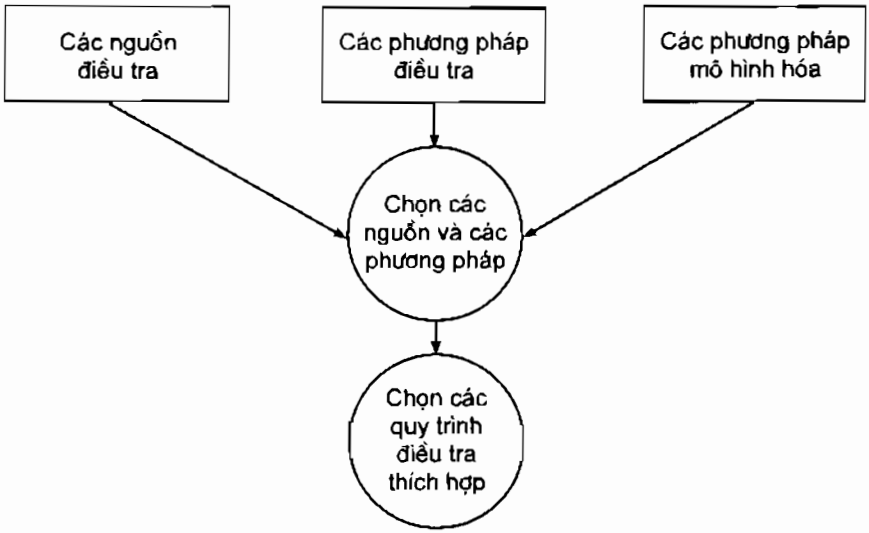
- Trung thực, khách quan, phản ánh đúng tình hình thực tại.
- Không bỏ sót thông tin.
- Các thông tin thu thập phải được đo đếm (số lượng, tần suất, độ chính xác, thời gian sống...).
- Không trùng lắp, nghĩa là phải tiến hành trong một trật tự, sao cho mỗi người được điều tra không bị nhiều người điều tra hỏi đi hỏi lại về một việc.
- Không gây cảm giác xấu hay phản ứng tiêu cực ở người bị điều tra: phải luôn luôn gợi mở, tế nhị, tuyệt đối không can thiệp vào công việc nội bộ cơ quan, hay làm tăng thêm các mâu thuẫn trong cơ quan.

d) Chiến lược điều tra

Một cuộc điều tra phải được thực hiện theo một chiến lược được cân nhắc kỹ càng từ trước. Một chiến lược bao gồm các yếu tố sau (xem Hình II.1):

- Các nguồn thông tin điều tra.
- Các phương pháp áp dụng cho mỗi nguồn thông tin điều tra.
- Các quy trình điều tra thích hợp.

Dưới đây, ta lần lượt đề cập đến các yếu tố kể trên.



Hình II. 1 Triển khai một chiến lược điều tra.

2. Các nguồn điều tra

Có nhiều nguồn thông tin trong hệ thống có thể được khai thác cho mục đích điều tra. Mỗi nguồn thường cung cấp các loại thông tin khác nhau và đòi hỏi các phương pháp khai thác khác nhau. Sau đây là một số nguồn thường gặp và các phương pháp thích hợp đối với mỗi nguồn đó.

a) Các người dùng hệ thống

Các người dùng hệ thống (những nhân viên, cán bộ trong cơ quan cũng như các khách hàng, đối tác ngoài cơ quan) là nguồn thông tin cần được điều tra đầu tiên. Từ các người dùng ta có thể tìm hiểu được sự hoạt động của hệ thống hiện tại, xác định được các mục tiêu và yêu cầu của mỗi người dùng. Phương pháp điều tra thường dùng ở đây là phỏng vấn và đôi khi là phiếu điều tra (sẽ đề cập ở dưới).

b) Các sổ sách, tài liệu

Các loại sổ sách, tài liệu, tệp máy tính... thường là nguồn thông tin để điều tra về các loại dữ liệu, luồng dữ liệu và giao dịch. Phương pháp khai thác nguồn này là đầu tiên lập một danh sách các tài liệu qua sự tìm hiểu từ các người dùng, rồi sau đó nghiên cứu từng tài liệu để phát hiện các dữ liệu cơ bản và các dữ liệu cấu trúc. Vào lúc này, cần chú ý phát hiện các dữ liệu trùng lắp hoặc sự thiếu nhất quán trong tên gọi để đảm bảo rằng không có dữ liệu nào xuất hiện dưới hai tên khác nhau.

c) Các chương trình máy tính

Các chương trình máy tính có thể được dùng để xác định các chi tiết về các cấu trúc dữ liệu và các quá trình xử lý. Phương pháp tìm hiểu ở đây là đọc kỹ chương trình hoặc tài liệu kèm theo và đôi khi có thể cho chạy chương trình với các dữ liệu kiểm chứng để thấy nó làm việc ra sao và có gì sai hỏng với giao diện người dùng hiện tại.

d) Các tài liệu mô tả quy trình, chức trách

Đó là những tài liệu quy định các quy trình làm việc và chức trách của các cán bộ nhân viên trong một cơ quan. Chúng có thể được dùng để người điều tra hiểu thêm chi tiết về công việc của các người dùng. Các chi tiết như vậy là quan trọng trong bước thiết kế chi tiết sau này. Phương phap khai thác là đọc tài liệu để thu gom các chi tiết có ích cho bước thiết kế về sau.

đ) Các thông báo

Các loại thông báo (như chứng từ giao dịch, giấy nhắc nợ...) là nguồn điều tra cho phép tìm hiểu các loại đầu ra cần thiết với các người dùng. Chúng có thể sử dụng làm căn cứ cho các cuộc phỏng vấn người dùng, để xác định còn chăng ở người dùng các đòi hỏi về các đầu ra mới nào nữa không.

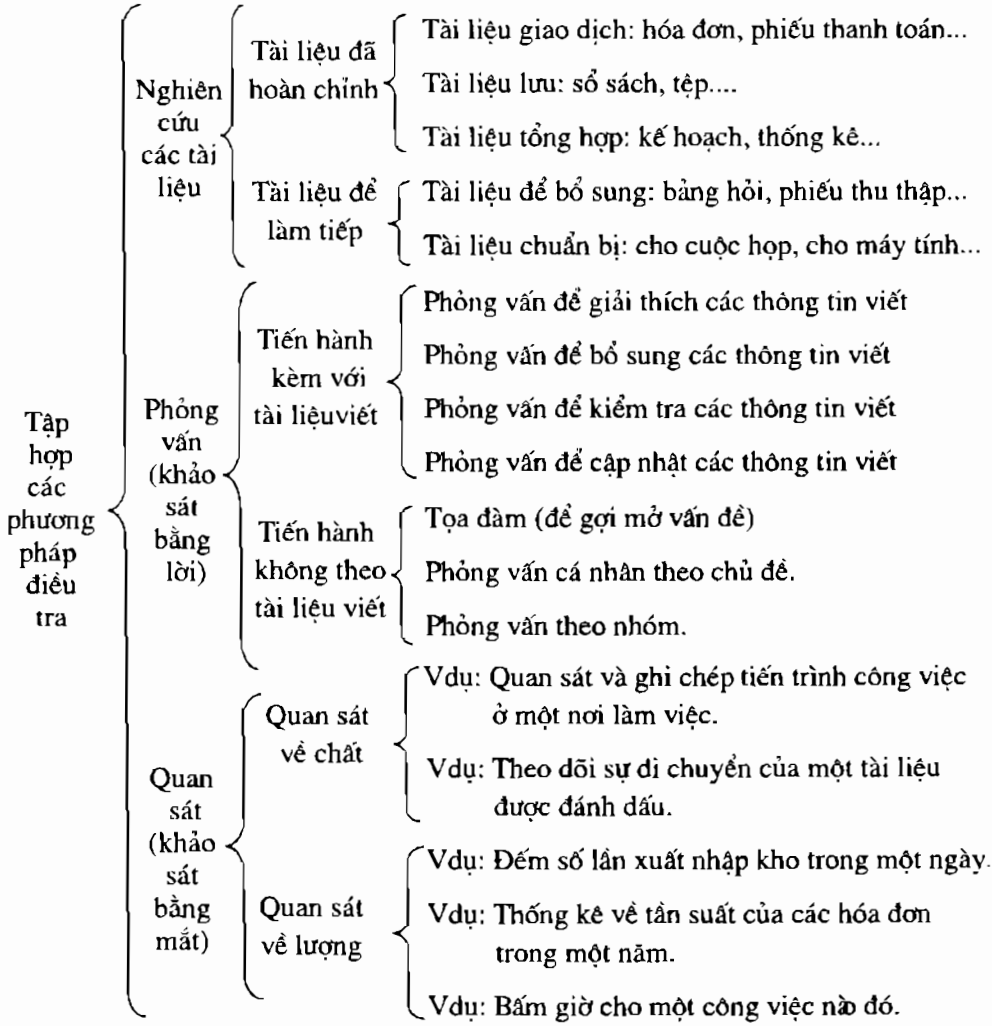
Đương nhiên, không một loại nguồn điều tra kể trên nào, riêng chúng có thể cung cấp đủ các thông tin cần thiết cho người điều tra. Cho nên nếu xuất phát từ một hệ thống hiện thời, thì ta phải tiến hành thu gom các thông tin về một cơ quan từ nhiều - nếu không phải là tất cả - các nguồn điều tra nói trên. Một quy trình điều tra sẽ xác định tiến trình khai thác các nguồn điều tra đó phải diễn ra như thế nào (xem mục 4 ở dưới).

3. Các phương pháp điều tra

Có thể tóm tắt và sắp xếp các phương pháp điều tra thường dùng như trong Bảng II.1. Dưới đây ta đề cập thêm về mỗi phương pháp đó.

a) Nghiên cứu tài liệu viết

Có thể nói đây là một sự quan sát gián tiếp, bởi vì đây cũng là sự khảo sát bằng mắt, nhưng không phải ở hiện trường, mà trên các tài liệu viết. Thông qua việc nghiên cứu các loại chứng từ giao dịch như hóa đơn, phiếu thanh toán, hoặc các loại sổ sách, các tệp máy tính, các tài liệu tổng hợp như kế hoạch, thống kê, biên bản, nghị quyết... ta có thể thu thập được nhiều loại thông tin, từ các hoạt động chung của cơ quan, đến các dữ liệu cơ bản, các dữ liệu cấu trúc. Nghiên cứu tài liệu thường kết hợp với phỏng vấn ở mức thấp (mức thao tác, thừa hành) để chi tiết hóa mô hình của hệ thống.



Bảng II.1. Phân loại các phương pháp điều tra

b) Quan sát

Đó là cách theo dõi (bằng mắt) tại hiện trường, nơi làm việc, một cách thụ động.

Việc quan sát thường đòi hỏi khá nhiều thời gian. Hơn nữa quan sát tỉ mỉ từng chi tiết nói chung không phải là phương pháp hữu hiệu để thu thập các thông tin cần thiết cho việc phát triển hệ thống máy tính. Một hệ thống mới thường sẽ làm thay đổi cung cách và các chi tiết thao tác, khiến cho các cung cách làm việc cũ không còn mấy ý nghĩa. Một hạn chế lớn nữa là người bị quan sát thường cảm thấy khó chịu, và họ thường thay đổi cách hành động khi bị quan sát - thay đổi theo chiều hướng không tốt.

Tuy nhiên kết hợp quan sát với phỏng vấn ngay tại nơi làm việc là một cách làm rất có hiệu quả.

c) Phỏng vấn

Đó là cách làm việc tay đôi hoặc theo nhóm, trong đó người điều tra đưa ra các câu hỏi và chắt lọc lấy các thông tin cần thiết qua các câu trả lời của các người được điều tra. Có thể nói đây là phương pháp cơ bản cho mọi cuộc điều tra.

Có hai loại câu hỏi:

- Câu hỏi mở, là câu hỏi mà số khả năng trả lời là rất lớn, người hỏi chưa hình dung hết được. Câu hỏi mở là có ích khi người hỏi chưa có ý định rõ ràng, muốn hỏi để thăm dò, để gợi mở vấn đề, và người trả lời phải là người có hiểu biết rộng (cán bộ lãnh đạo chẳng hạn).
- Câu hỏi đóng, là câu hỏi mà các phương án trả lời có thể dự kiến sẵn, chỉ cần khẳng định đó là phương án nào. Câu hỏi đóng là có ích khi ta đã có chủ định điều tra và cần biết rõ các chi tiết.

Việc sắp xếp các câu hỏi cho hợp lý, phù hợp với chủ định điều tra và khả năng của người trả lời, là điều phải cân nhắc kỹ. Trật tự các câu hỏi có thể là:

- thu hẹp dần: ban đầu là những câu hỏi khái quát, về sau hỏi tập trung vào một chủ điểm, một chi tiết nhất định.
- mở rộng dần: ban đầu đề cập một vài việc cụ thể nào đó, rồi mở rộng dần phạm vi được đề cập.
- thắt rồi mở: tập trung dần vào một chủ điểm, rồi lại bung ra để khuếch trương kết quả thu được.

Tuy nhiên, trong bất cứ trường hợp nào, thì sự dẫn dắt của người phỏng vấn qua các câu hỏi đưa ra cũng không được thể hiện một sự áp đặt, một định kiến chủ quan.

Thiết lập một quan hệ hợp tác trong quá trình phỏng vấn là điều thiết yếu. Người được phỏng vấn luôn luôn có mặc cảm. Bởi vậy ngay trong buổi đầu gặp gỡ, ta phải cố gắng thiết lập ngay một quan hệ đồng cảm, để hai bên cùng chia sẻ một động cơ chung, vì lợi ích của cơ quan. Người phỏng vấn phải luôn luôn tỏ ra biết chú ý lắng nghe. Trung thực là đối sách tốt nhất. Phải luôn luôn biểu lộ sự tin cậy, thiện cảm, tôn trọng.

d) Phiếu điều tra

Có thể nói đây là một hình thức phỏng vấn không giáp mặt. Các câu hỏi được liệt kê trong một mẫu điều tra, và người được điều tra ghi các trả lời của mình vào mẫu đó.

Việc sử dụng các loại câu hỏi cũng như trật tự các câu hỏi ở đây cũng giống như khi phỏng vấn trực tiếp. Tuy nhiên trật tự các câu hỏi trên phiếu điều tra chỉ có ý nghĩa về mặt sắp xếp vấn đề, mà ít có tác dụng dẫn dắt tư duy của người được hỏi, vì người được hỏi ở đây có thể đọc qua một lượt các câu hỏi, rồi tùy tiện trả lời các câu hỏi theo bất cứ trật tự nào mà họ muốn (thậm chí không trả lời một số câu hỏi).

So với phỏng vấn, thì chỗ yếu chính của phiếu điều tra là thiếu sự giao tiếp giữa người hỏi và người đáp. Trong phỏng vấn, nhiều khi ngôn ngữ nói không phải là công cụ duy nhất để truyền đạt. Nét mặt, cử chỉ, dáng điệu có thể cho ta biết một người đang nghĩ gì, những điều mà họ không muốn nói ra bằng lời.

Ngược lại, phiếu điều tra lại có chỗ mạnh là có thể mở rộng diện điều tra, và ít tốn kém.

4. Các quy trình điều tra

Một quy trình điều tra là một kế hoạch xác định việc khai thác các nguồn điều tra cần được tiến hành theo trật tự nào, với các phương pháp nào và nhằm thu thập những thông tin nào.

Mục đích của điều tra là để đi tới hiểu rõ hệ thống hiện tại. Mà như ta đã thấy ở Chương I, thì "Hiểu tức là mô hình hóa". Khi tiến hành điều tra để thu gom các thông tin về hệ thống, thì dần dần từng bước người điều tra sẽ hình dung ra hệ thống ngày một rõ rệt hơn. Nói cách khác, anh ta dần dần hình thành một mô hình của hệ thống. Mô hình này chỉ có thể hình thành một cách dần dần: ban đầu là đại lược, về sau sẽ tăng thêm chi tiết; ban đầu là chưa chính xác, về sau sẽ được chỉnh sửa lại cho chính xác hơn; ban đầu là chưa đầy đủ, về sau sẽ được bổ sung cho đầy đủ hơn.

Chính vì các lẽ đó, mà khi vạch ra một quy trình điều tra, ta phải tuân thủ ba nguyên tắc sau:

a) Quy trình điều tra phải hỗ trợ một cách đắc lực nhất cho phương pháp mô hình hóa

Trong giai đoạn nghiên cứu sơ bộ, thì mô hình được dùng thường là mô hình diễn tả bằng ngôn ngữ tự nhiên, kết hợp với một số sơ đồ, biểu đồ dễ hiểu, để dễ dàng trao đổi, thảo luận với người dùng. Chẳng hạn ở mục (5) dưới đây, ta sẽ đưa ra một số tiêu chuẩn phân loại các thông tin thu thập được và dựa trên các tiêu chuẩn phân loại đó, ta đưa ra một cách trình bày các kết quả điều tra. Thực chất bản trình bày đó là một mô hình của hệ thống. Với cách mô hình hóa đó thì quy trình điều tra phải được thiết kế để nhằm khai thác thuận lợi nhất các loại thông tin được đề cập trong mô hình. Sau này, ở giai đoạn phân tích hệ thống, mô hình sử dụng có thể là biểu đồ luồng dữ liệu, là mô hình thực thể/ liên kết, thì bấy giờ sự điều tra lại phải nhằm vào các thông tin ở mức lôgic, mà bỏ qua các thông tin ở mức vật lý.

b) Quy trình điều tra phải được tiến hành trên xuống

Bởi lẽ mô hình phải luôn luôn hình thành dần dần từ đại thể tới chi tiết cho nên điều tra cũng nên tiến hành từ trên xuống. Ta thường phân biệt ba mức điều tra kể từ trên xuống:

- mức quyết định lãnh đạo (ban giám đốc, hội đồng quản trị, các chuyên gia, cố vấn),
- mức điều phối, quản lý (các trưởng phòng, ban, cửa hàng, phân xưởng...),
- mức thao tác, thừa hành (thủ kho, thủ quỹ, kế toán viên, thư ký...).

Việc điều tra thường bắt đầu bằng vài cuộc tọa đàm với các vị lãnh đạo, qua đó ta hình dung được một cách bao quát sự hoạt động của cả cơ quan, trong một thời hạn lâu dài. Chính các vị lãnh đạo sẽ giúp ta biết nên tìm hiểu kỹ thêm ở các cán bộ cấp dưới nào. Việc trao đổi với các cán bộ cấp dưới này, đặc biệt là việc quan sát, phỏng vấn các cán bộ thừa hành sẽ giúp ta bổ sung thêm nhiều chi tiết cụ thể làm cho mô hình của hệ thống trở nên rõ nét hơn.

c) Quá trình điều tra lại phải được tiến hành lặp đi lặp lại

Một lần đi từ trên xuống chưa đủ. Thường khi đi vào chi tiết và cụ thể, ta lại phát hiện ra cái gì đó còn thiếu sót hay chứa đựng mâu thuẫn trong bức tranh đại thể của cơ quan. Điều đó đòi hỏi ta lại phải trở lại làm việc với cấp

trên để được giải thích rõ hơn, để rồi lại đi xuống dưới đào sâu thêm về những điều mới lĩnh hội. Cứ thế, phải lặp đi lặp lại nhiều lần, qua đó chỉnh sửa lại mô hình cho hoàn thiện hơn, đầy đủ hơn.

5. Phân loại và biên tập các thông tin điều tra

Các thông tin thu thập được qua quá trình điều tra cần phải được rà soát, phân loại và biên tập lại.

Sự phân loại các thông tin điều tra có thể dựa vào các tiêu chuẩn phân biệt như sau:

- Hiện tại/ Tương lai: Nhiều thông tin thu được là thông tin phản ánh thực trạng hiện tại, song cũng không ít thông tin chỉ là phản ánh sự mong muốn chủ quan hoặc nhu cầu trong tương lai. Cả hai loại đều có ích, song phải phân biệt rõ ràng.

- Nội bộ/ Môi trường: Có những thông tin phản ánh tình trạng nội bộ của hệ thống, và có những thông tin đề cập về môi trường, hoàn cảnh của hệ thống. Sự xác định ranh giới của hệ thống, tức là sự phân biệt nội bộ hệ thống với môi trường của hệ thống nhiều khi rất mơ hồ. Cần thảo luận kỹ với cơ quan chủ quản để xác định dứt khoát ranh giới đó.

- Tĩnh/ Động/ Biến đổi: Có những thông tin phản ánh tình trạng tĩnh tại, ổn định của hệ thống như là thông tin mô tả cơ cấu tổ chức của cơ quan, các tình trạng thiết bị, nhà xưởng, nhân sự, các loại sổ sách, các tệp... Có những thông tin phản ánh động thái của hệ thống, chẳng hạn động thái trong không gian như đường đi của các tài liệu trong quá trình xử lý, trong thời gian như thời gian xử lý, các hạn định chuyển giao thông tin... Có những thông tin thu thập được lại đề cập cách biến đổi, chế biến các dữ liệu, như là các quy tắc quản lý, các công thức tính toán, các điều kiện để khởi động, các quy trình xử lý...

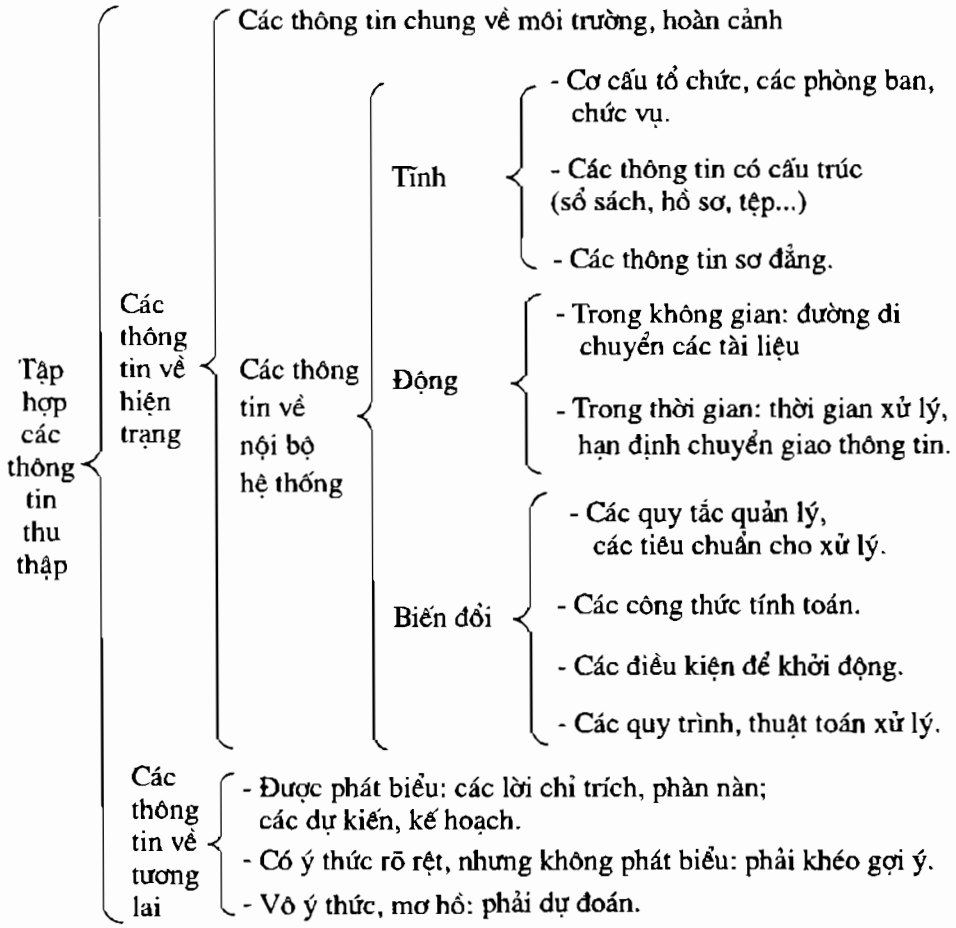
Sự phân biệt các loại thông tin điều tra như trên cho phép ta sắp xếp lại các thông tin đó một cách có hệ thống. Chẳng hạn có thể sắp xếp chúng như trong Bảng II.2. Đương nhiên đó chỉ là một phương án. Có thể tùy thuộc vào đặc thù cụ thể của hệ thống đang nghiên cứu mà ta có thể đưa ra các phương án sắp xếp khác cho nổi bật các điểm cốt yếu hơn.

Rốt cuộc thì sau khi rà soát, phân loại, biên tập lại các thông tin thu thập được trong quá trình điều tra, ta hình thành được một bản trình bày mô tả các

thành phần và sự hoạt động của hệ thống. Bản trình bày này có thể kèm theo một số đồ thị, bảng biểu minh họa (chẳng hạn lưu đồ hệ thống, các bảng quyết định...). Bản trình bày dùng để trao đổi trong nhóm những người triển khai hệ thống cũng như để trao đổi với các người dùng và cơ quan chủ quản hệ thống. Nó cũng là một tư liệu trong hồ sơ của giai đoạn tìm hiểu nhu cầu.

6. Phê phán hiện trạng

Kết thúc quá trình khảo sát điều tra hiện trạng là sự chỉ trích các yếu kém của hiện trạng đó.



Bảng II.2. Sắp xếp các thông tin điều tra

Đây là một công việc khó khăn, tế nhị. Đành rằng hệ thống cũ có bộc lộ nhiều điều bất cập thì cơ quan chủ quản mới đặt vấn đề xây dựng một hệ thống mới thay thế cho nó. Tuy nhiên phanh phui các điểm yếu kém đến tận từng chi tiết vẫn dễ gây cảm giác khó chịu và phản ứng không hay ở người trong cuộc. Song nếu không nhận định rõ các yếu kém của hiện trạng và thỏa thuận công khai với người dùng về các yếu kém đó, thì khó xác định được mục tiêu của dự án phát triển hệ thống mới, thậm chí khó biện minh cho sự cần thiết tồn tại của dự án này. Vậy đây là việc không thể trốn tránh, chỉ có điều là phải thực hiện nó với sự thận trọng, khiêm tốn và tế nhị mà thôi.

Có thể có ba loại yếu kém:

- Sự thiếu, vắng: thiếu thông tin cho xử lý, thiếu nhân lực, thiếu phương tiện, bỏ sót công việc đáng làm.
- Sự kém hiệu lực: thể hiện trên nhiều mặt:
  - cơ cấu tổ chức bất hợp lý;
  - phương pháp xử lý không chặt chẽ;
  - lưu chuyển giấy tờ bất hợp lý, thường là vòng vèo, quá dài;
  - giấy tờ, sổ sách trình bày kém, thiếu thông tin, cấu trúc dở;
  - để xảy tình trạng ùn tắc, quá tải, người làm quá mệt mỏi.
- Sự tốn kém:
  - chi phí quá cao;
  - lãng phí vô ích.

Cùng với các yếu kém của hiện trạng, ta cũng nên đính kèm các yêu cầu nảy sinh cho tương lai (đã nói ở trên):

- các nhu cầu về thông tin chưa được đáp ứng,
- các mong muốn, nguyện vọng của nhân viên,
- các dự kiến, kế hoạch phát triển từ phía lãnh đạo.

Để kết thúc phần khảo sát hiện trạng, ta hãy xét một thí dụ. Thí dụ này, được gọi là Thí dụ QL (quản lý), sẽ cùng với một thí dụ nữa, gọi là Thí dụ ĐK (điều khiển) được đưa ra sau này, sẽ là hai thí dụ xuyên suốt, minh họa cho mọi bước phân tích và thiết kế hệ thống.

Thí dụ QL: Hệ cung ứng vật tư (CƯVT) ở nhà máy Z. Nhà máy Z là một nhà máy cơ khí lớn. Gần đây bộ phận cung ứng vật tư sản xuất của nhà máy tỏ ra bất cập, không đáp ứng kịp thời các nhu cầu sản xuất tại các phân xưởng. Vì vậy có yêu cầu cải tiến quản lý ở bộ phận này.

(1) *Nhiệm vụ cơ bản*:

Khi các phân xưởng có yêu cầu vật tư, thì bộ phận CƯVT phải thực hiện mua hàng ở các nhà cung cấp, đưa về đáp ứng kịp thời cho các phân xưởng, không để xảy ra các sai sót về hàng nhận và tiền trả.

(2) *Cơ cấu tổ chức và sự phân công trách nhiệm*:

Thực ra bộ phận cung ứng vật tư gồm ba tổ, hoạt động tương đối độc lập với nhau:

- Tổ thứ nhất đảm nhiệm việc *đặt hàng* dựa trên các dự trù vật tư của các phân xưởng. Tổ này có sử dụng một máy tính nhỏ, trên đó có một hệ chương trình gọi là hệ Đặt hàng (ĐH) trợ giúp các việc chọn người cung cấp, làm đơn hàng và theo dõi sự hoàn tất của đơn hàng.

- Tổ thứ hai đảm nhiệm việc *nhận và phát hàng*. Tổ này cũng có một máy tính nhỏ, trên đó có một hệ chương trình gọi là hệ Phát hàng (PH) trợ giúp các việc ghi nhận hàng về và làm thủ tục phát hàng cho các phân xưởng.

- Tổ thứ ba gọi là tổ *Đối chiếu và kiểm tra*. Sở dĩ có tổ này là vì hai máy tính ở hai tổ nói trên là không tương thích cho nên không nối ghép được với nhau. Vì vậy các thông tin về đặt hàng và nhận hàng quản lý ở hai máy tính đó là hoàn toàn bị tách rời và do đó hàng về mà không xác định được là hàng cho phân xưởng nào. Chính tổ đối chiếu sẽ lấy các thông tin của các đợt đặt hàng và của các đợt nhận hàng từ hai tổ nói trên về, khớp lại để tìm ra phân xưởng có hàng, giúp cho tổ thứ hai thực hiện việc phát hàng. Tổ đối chiếu còn có nhiệm vụ phát hiện các sai sót về hàng và tiền để khiếu nại với các nhà cung cấp nhằm chỉnh sửa lại cho đúng. Tổ đối chiếu làm việc hoàn toàn thủ công.

(3) *Quy trình xử lý và các dữ liệu xử lý*

Qua điều tra khảo sát, ta thấy quy trình làm việc, cùng các loại chứng từ giao dịch sử dụng trong quy trình đó như sau:

Khi có nhu cầu vật tư, một phân xưởng sẽ lập một bản *dự trù* gửi cho tổ Đặt hàng, trong đó có các mặt hàng được yêu cầu, với các số lượng yêu cầu tương ứng. Tổ đặt hàng trước hết chọn nhà cung cấp để đặt mua các mặt hàng nói trên. Muốn thế, nó dùng máy tính để tìm các thông tin về các người cung cấp được lưu trong tệp NGCCẤP. Sau đó nó thương lượng trực tiếp với người cung cấp được chọn (gặp mặt hoặc qua điện thoại). Sau khi đã thỏa thuận, dùng hệ chương trình ĐH để in một *đơn hàng*. Các thông tin trên đơn hàng được lưu lại để theo dõi trong tệp ĐƠN HÀNG, còn đơn hàng in ra thì gửi tới người cung cấp. Để tiện theo dõi, người ta áp dụng nguyên tắc: mỗi khoản đặt hàng trên một đơn hàng giải quyết trọn vẹn (nghĩa là không tách, không gộp) một khoản yêu cầu về một mặt hàng trên một bản dự trù. Tuy nhiên một đơn hàng, gồm nhiều khoản, có thể đáp ứng yêu cầu của nhiều dự trù khác nhau. Ngược lại các khoản yêu cầu trên một bản dự trù lại có thể được phân bổ lên nhiều đơn hàng khác nhau, gửi đến các nhà cung cấp khác nhau. Lại chú ý rằng đơn hàng gửi tới nhà cung cấp không chứa thông tin về phân xưởng đã dự trù hàng đặt. Vì vậy cần lưu mối liên hệ giữa các bản dự trù của các phân xưởng với các đơn hàng đã được phát đi trong một tệp gọi là tệp DT-ĐH, ở đó đặt liên kết mỗi Số hiệu dự trù và mỗi Số hiệu mặt hàng với một Số hiệu đơn hàng.

Nhà cung cấp, căn cứ trên đơn đặt hàng, sẽ chuyển hàng đến nhà máy, kèm *phiếu giao hàng*. Tổ nhận và phát hàng tiếp nhận hàng đó. Hàng thì cất tạm vào một kho (có nhiều kho), còn thông tin trên phiếu giao hàng cùng địa điểm cất hàng được lưu vào máy tính, trong tệp NHẬN HÀNG. Trên phiếu giao hàng, mỗi mặt hàng được giao đều có ghi rõ Số hiệu đơn hàng đã đặt mặt hàng đó (số lượng giao có thể là chưa đủ như số lượng đặt). Tuy vậy, như thế vẫn chưa rõ hàng đó là do phân xưởng nào yêu cầu để có thể phát hàng về phân xưởng ngay được.

Để giải quyết mắc míu này, hàng tuần tổ nhận hàng sử dụng hệ chương trình PH, in ra một *danh sách Nhận hàng* trong tuần, gửi cho tổ Đối chiếu, với nội dung như sau:

SH giao hàng - Tên NgCCấp - SH Mặt hàng - Số lượng nhận - SH đơn hàng.

Mặt khác, cũng hàng tuần, tổ Đặt hàng sử dụng hệ chương trình ĐH, in ra một *danh sách Đặt hàng* trong tuần, gửi cho tổ Đối chiếu, với nội dung như sau:

SH đơn hàng - Tên NgCCấp - SH mặt hàng - Số lượng đặt - SH Dự trù - Tên PX

Tổ Đối chiếu khớp hai danh sách này, tìm ra SH đơn hàng và SH Mặt hàng chung, và từ đó xác định được lượng hàng nào là cần phát về cho phân xưởng nào. Danh sách các địa chỉ phát hàng được lập và gửi lại cho tổ Nhận và Phát hàng, để tổ này chuyển hàng kèm *phiếu phát hàng* cho các phân xưởng.

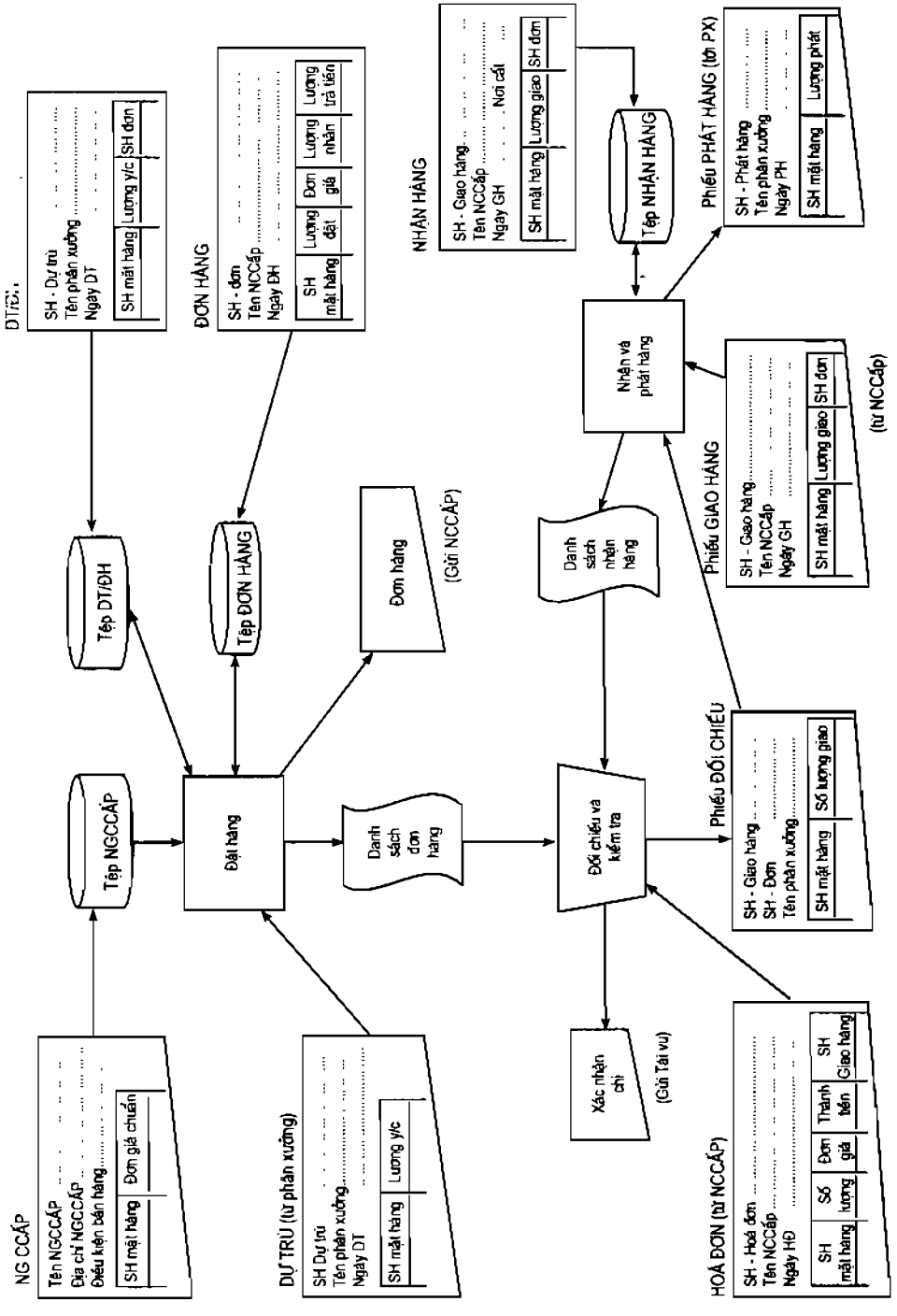
Tổ Đối chiếu và kiểm tra còn có nhiệm vụ tiếp nhận *hóa đơn* từ nhà cung cấp gửi đến, đối chiếu nó với hàng đã nhận, nếu chính xác thì xác nhận chi lên hóa đơn và gửi cho bộ phận thanh toán (thuộc phòng Tài vụ) để làm thủ tục trả tiền. Nếu phát hiện có sự không ăn khớp giữa hàng đặt - hàng nhận và tiền phải trả, thì tổ Đối chiếu và kiểm tra khiếu nại với nhà cung cấp để chỉnh sửa lại. Việc kiểm tra thường có khó khăn, vì lắm khi nhà cung cấp thiếu hàng, chưa đáp ứng đủ ngay mà còn nợ lại một phần để giao sau. Còn về phía nhà máy, có khi chưa đủ tiền để trả đủ theo hóa đơn, mà còn nợ lại một phần để trả sau. Mặt khác, thì tổ Đặt hàng lại cũng muốn biết đơn hàng do mình phát ra là đã hoàn tất hay chưa, cho nên tổ này đã yêu cầu bộ phận thanh toán mỗi khi trả tiền cho nhà cung cấp thì gửi cho tổ một bản ghi trả tiền. Thông tin trả tiền này được cập nhật vào tệp ĐƠN HÀNG, nhờ đó biết đơn hàng nào là đã hoàn tất.

Quy trình làm việc của ba tổ như trên, cùng với các loại dữ liệu chuyển giao và lưu trữ có thể tóm tắt trong một lưu đồ hệ thống như trong Hình II. 2.

(4) *Phê phán hiện trạng*

Có thể nêu ra các yếu kém sau đây của hệ thống trên:

- Thiếu: Thiếu một kho hàng dự trữ các mặt hàng thông dụng và rẻ tiền. Đầu tư cho một kho như thế không tốn kém mấy mà đáp ứng ngay được cho 60% yêu cầu vật tư hàng ngày.
- Kém:
  - Chu trình quá lâu, do khâu chờ đợi địa chỉ phát hàng.
  - Kiểm tra không chặt, để xảy ra sai sót hàng - tiền luôn.
- Tốn: Tốn nhân lực ở khâu đối chiếu và kiểm tra bằng tay.



Hình II. 2 Lưu đồ hệ thống hệ CƯVT nhà máy Z

§2 XÁC LẬP VÀ KHỞI ĐẦU DỰ ÁN

Sau khi đã thấy rõ các yếu kém của hiện trạng và các yêu cầu phát triển trong tương lai, thì việc xây dựng một hệ thống mới thay thế cho hệ thống cũ được bàn bạc để nhất trí. Từ đó cần xác lập và khởi đầu một dự án xây dựng hệ thống mới đó. Cần thực hiện các việc sau đây:

- Xác định phạm vi và các hạn chế cho dự án.
- Xác định các mục tiêu và ưu tiên cho dự án.
- Đề xuất một số giải pháp thô và phân tích tính khả thi của chúng để chọn một giải pháp.
- Lập kế hoạch triển khai dự án.

1. Xác định phạm vi và các hạn chế

Trước hết cần có sự thỏa thuận minh bạch giữa cơ quan chủ quản và các người phát triển hệ thống về phạm vi, ranh giới của các vấn đề đặt ra với dự án. Phạm vi đó có thể bao trùm cả cơ quan hay chỉ đụng chạm một vài bộ phận nhỏ của cơ quan, bao quát công tác quản lý toàn diện hay chỉ giải quyết một vài công tác quản lý riêng biệt nào đó.

Cần phân biệt ba loại kích cỡ của cơ quan chủ quản, mà độ phức tạp của chúng là rất khác nhau:

- Cơ quan lớn, cỡ quốc gia hay quốc tế gồm nhiều đơn vị phân bố trên địa bàn rộng. Chẳng hạn một ngân hàng tập trung, một tập đoàn bảo hiểm, một tổng công ty (điện lực, bưu điện, đường sắt, hàng không...), một cơ quan hành chính cấp bộ v.v...
- Cơ quan trung bình, thường chỉ là một đơn vị, nhưng có nhiều chi nhánh, văn phòng phân tán về mặt địa dư.
- Xí nghiệp nhỏ và vừa, ở đó người giám đốc xí nghiệp có hiểu biết tường tận về xí nghiệp của mình.

Trong một cơ quan, dù là lớn hay nhỏ, thì hệ thống quản lý thường bao gồm nhiều hệ con (như quản lý nhân sự, quản lý tài chính, quản lý vật tư, quản lý sản xuất v.v...), và mỗi hệ con lại gồm nhiều hệ nhỏ hơn (chẳng hạn quản lý kho, xử lý đơn hàng v.v...). Thông thường thì ta cần có một nghiên cứu tổng

quát về chiến lược tin học hóa công tác quản lý của cơ quan, qua đó xác định các hệ con, cùng các loại giao diện giữa chúng (giao diện càng đơn giản và càng lỏng lẻo càng tốt), rồi sau đó chọn một số hệ con đó đưa vào phạm vi giải quyết của dự án (gọi là một ứng dụng).

Dù sao thì ranh giới của hệ thống ít khi có thể xác định được thật rành mạch. Sau này qua các bước phân tích và thiết kế sâu có thể cần trở lại chỉnh sửa lại ranh giới đó đôi chút.

Các hạn chế đối với dự án, do cơ quan chủ quản đề xuất tùy theo khả năng của mình, cũng cần được trao đổi kỹ. Có thể có ba loại hạn chế:

- Hạn chế về nhân lực: các người có thể tham gia dự án cũng như người dùng hệ thống sau này (số lượng, trình độ).
- Hạn chế về thiết bị, kỹ thuật: các khả năng về thiết bị và kỹ thuật có thể đáp ứng.
- Hạn chế về tài chính: mức độ đầu tư và chi phí dành cho dự án.

2. Xác định các mục tiêu và ưu tiên cho dự án

Thông thường thì một hệ thống thông tin được xây dựng là nhằm vào các mục đích sau:

- Mang lại lợi ích nghiệp vụ: tăng khả năng xử lý; đáp ứng yêu cầu nghiệp vụ một cách tin cậy, chính xác, an toàn, bí mật.
- Mang lại lợi ích kinh tế: giảm biên chế cán bộ, giảm chi phí hoạt động; tăng thu nhập, hoàn vốn nhanh.
- Mang lại lợi ích sử dụng: nhanh chóng, thuận tiện.
- Khắc phục các khiếm khuyết của hệ thống cũ; hỗ trợ các chiến lược phát triển lâu dài; đáp ứng các ưu tiên, ràng buộc và hạn chế đã được áp đặt.

Tuy nhiên khi vạch các mục tiêu cần đạt được cho một dự án xây dựng hệ thống thông tin, thì ta nên chọn những mục tiêu tương đối cụ thể để sau này có thể kiểm điểm sự hoàn tất của dự án một cách dễ dàng.

Ngoài các mục tiêu, thì ta còn phải ghi nhận một số ưu tiên mà sau này khi thực hiện dự án, ta luôn luôn phải xem xét đến trong chọn lựa các quyết định. Các ưu tiên đó có thể do cơ quan chủ quản và người dùng đề xuất, cũng có thể

nảy sinh từ một hoàn cảnh khách quan nào đó. Có thể đó là những ràng buộc về kiến trúc hệ thống, về sử dụng thiết bị, về chi phí, địa điểm, thời hạn thực hiện v.v... Các ràng buộc này không nhất thiết phải tuân thủ một cách ngặt nghèo (như các hạn chế đã nói ở mục 1) mà chỉ là những điểm nên cố gắng chọn lựa khi ta đứng trước nhiều phương án chọn lựa tương đương.

Thí dụ QL (tiếp theo): Đối với việc xây dựng hệ Cung ứng vật tư trong nhà máy Z, thì ban đầu, một cách đại thể, có thể đưa ra hai mục tiêu như sau:

- Rút ngắn thời gian giữa dự trù và phát hàng;
- Loại trừ các sai sót trong việc xử lý các đơn hàng.

Tuy nhiên, để cụ thể hơn, ta phân tích các mục tiêu trên, bằng cách đặt câu hỏi: "Làm thế nào để đạt được các mục tiêu đại thể đó?". Có nhiều khả năng chọn lựa, chẳng hạn ta chọn các mục tiêu sau:

- Đưa thêm một kho hàng dự trữ để có thể giải quyết các yêu cầu về các mặt hàng thông thường ngay tại kho;
- Cải tiến cách xác định địa chỉ phát hàng để tránh chờ đợi lâu;
- Cải tiến và thống nhất việc kiểm tra, để tránh các sai sót giữa các hàng đặt, hàng nhận và tiền trả.
- Chuyển công việc của tổ Đối chiếu và kiểm tra từ thực hiện thủ công sang thực hiện trên máy tính để có thể giảm biên chế cán bộ ở tổ này.

Mặt khác, ta ghi nhận một điểm ưu tiên mà phía cơ quan chủ quản mong muốn là: Cố gắng tận dụng hai máy tính đã có và các chương trình còn có thể dùng được trên hai máy tính đó.

3. Phác hoạ giải pháp và cân nhắc tính khả thi

Việc đưa ra giải pháp vào giai đoạn này, nghĩa là vào lúc mới chỉ tìm hiểu sơ bộ, khi chưa có một sự phân tích sâu sắc và một sự cân nhắc kỹ lưỡng, quả thật là có phần hơi sớm. Tuy nhiên việc đó không thể thoái thác vì mấy lý do:

- Trước hết là bên sử dụng không thể chờ đợi lâu hơn. Một khi người ta sắp phải quyết định đầu tư cho một dự án xây dựng hệ thống thông tin mới, mà lại chưa được nhìn thấy rõ một giải pháp có triển vọng và tin tưởng được, trái lại chỉ được nghe các lời hứa suông của bên xây dựng hệ thống, thì làm sao người ta có thể thực sự yên tâm được?

- Mặt khác, bên xây dựng hệ thống cũng cần có một sự xác định giải pháp như thế. Bởi vì một giải pháp giúp ta dự toán được mức đầu tư cho dự án (về phần cứng, phần mềm) và một giải pháp sẽ cho một định hướng tốt cho cả quá trình phát triển hệ thống tiếp theo.

Việc đưa ra một giải pháp như thế, tuy có hơi sớm, song lại có thể làm được, vì đây chỉ là giải pháp thô, nghĩa là nó chỉ bao gồm các quyết định đại lược, như là:

- Chức năng chính của hệ thống, đầu vào, đầu ra, các biện pháp chính để đáp ứng nhu cầu của người dùng;
- Kiến trúc tổng thể của hệ thống, bao gồm kiến trúc phần mềm (các hệ thống con, các môđun chính...) và kiến trúc phần cứng (mạng, các máy tính, các thiết bị khác...).

Cái chủ yếu của giải pháp là, tuy chưa chi tiết, song nó phải đủ rõ để thuyết phục được bên sử dụng về khả năng đáp ứng nhu cầu của nó và đủ rõ để chứng tỏ được tính khả thi.

Thường thì ban đầu, ta nên đề xuất nhiều giải pháp. Các giải pháp đó có thể ở những mức độ tự động hóa khác nhau, miễn là nó có mang lại một sự cải tiến quản lý tích cực:

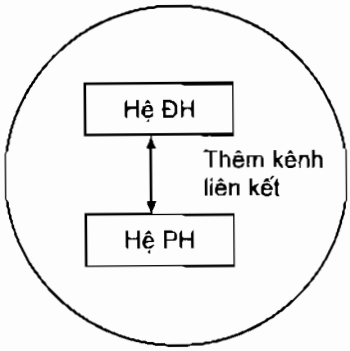
- Có thể đó chỉ là một sự tổ chức lại các hoạt động thủ công;
- Có thể đó là sự tự động hóa ở mức độ vừa phải: dùng máy tính trợ giúp cho một số công việc, không gây ra sự đảo lộn lớn về tổ chức cơ quan;
- Có thể đó là sự tự động hóa ở mức cao, kèm theo sự thay đổi đáng kể về cơ cấu tổ chức và cung cách làm việc của cơ quan.

Sau khi đã xác định một số giải pháp, ta phải đánh giá tính khả thi của từng giải pháp, để từ đó tiến hành cân nhắc, so sánh và thỏa thuận với bên sử dụng chọn lấy một giải pháp thỏa đáng nhất. Tính khả thi có thể cân nhắc trên các mặt sau:

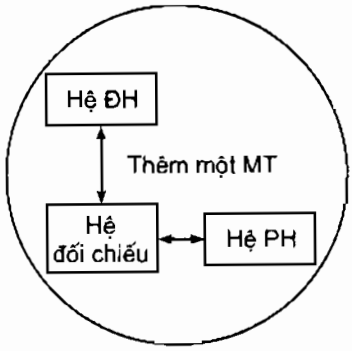
- Khả thi về nghiệp vụ: có đáp ứng các nhu cầu và nghiệp vụ của bên sử dụng không (cung cấp đúng các thông tin nghiệp vụ cần thiết vào đúng lúc yêu cầu và đến đúng nơi)?
- Khả thi về kỹ thuật: các yêu cầu về kỹ thuật và công nghệ của giải pháp có thể đáp ứng được không?

- Khả thi về kinh tế: chi phí cho giải pháp là có thể đáp ứng được không, có thỏa đáng so với lợi ích thu lại không?

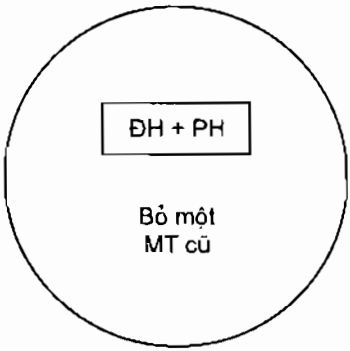
Thí dụ QL (tiếp theo): Với hệ CƯVT ở nhà máy Z, ta có thể hình dung năm giải pháp khác nhau như sau:



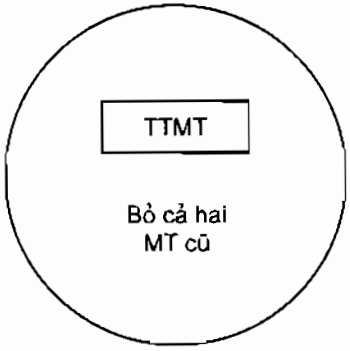
Giải pháp 1: Thiết lập 1 kênh liên kết giữa hai máy tính vốn có. Xóa bỏ tổ Đối chiếu và Kiểm tra.



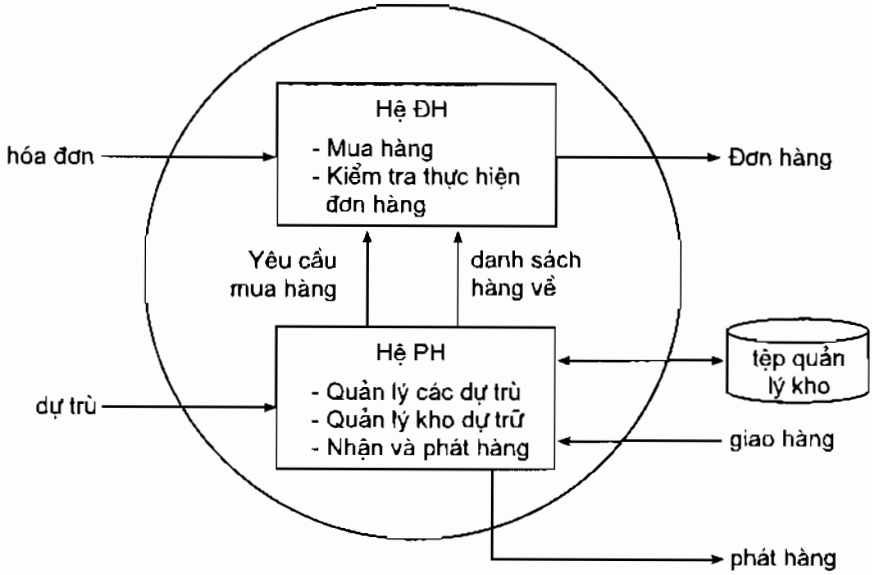
Giải pháp 2: Thêm một máy tính để giải quyết khâu Đối chiếu. Ba máy tính được kết nối.



Giải pháp 3: Gộp hệ ĐH vào hệ PH hay ngược lại. Bỏ một MT cũ. Bỏ tổ Đối chiếu và Kiểm tra thủ công



Giải pháp 4: Bỏ cả hai MT cũ, đưa bài toán vào TTMT chung của cơ quan. Tổ chức lại các phần việc thủ công.



Hình II. 3 Các giải pháp được đề xuất

Giải pháp 5: Bỏ tổ Đối chiếu và kiểm tra thủ công và giữ lại hai hệ ĐH và PH (trên hai máy tính cũ) với ít nhiều điều chỉnh nhiệm vụ: Hệ ĐH vẫn thực hiện việc đặt hàng, nay thêm việc kiểm tra thực hiện đơn hàng; hệ PH vẫn thực hiện việc nhận và phát hàng, nay thêm nhiệm vụ quản lý dự trù (do đó tự túc được việc tìm địa chỉ phát hàng) và thêm nhiệm vụ quản lý kho hàng dự trữ. Yêu cầu mua hàng và danh sách hàng về được chuyển bằng tay từ hệ PH sang hệ ĐH.

Cân nhắc tính khả thi và chọn lựa giải pháp:

- Các giải pháp 1 và 2 đều không khả thi về kỹ thuật, vì các máy tính cũ là không tương thích, cho nên không nối ghép được. Vả lại giải pháp 2 chuyển việc đối chiếu và kiểm tra vốn thực hiện bằng tay sang thực hiện trên máy tính mà không cải tiến lại quy trình xử lý thì cũng kém hiệu quả.

- Giải pháp 3 là có thể được, nếu máy tính cũ giữ lại là đủ mạnh. Tuy nhiên nếu hai máy tính là không tương thích đến mức không chạy được chương trình của nhau, thì như vậy ít ra ta cũng phải viết lại khoảng một nửa số chương trình (kém khả thi về kinh tế).

- Giải pháp 4 là rất có ích nếu thông tin về CƯVT còn được sử dụng ở một số bộ phận khác trong cơ quan. Tuy nhiên dường như CƯVT chỉ là một

công việc đóng khung trong một bộ phận nhỏ, không đáng phải đem hòa chung vào quy trình xử lý thông tin tổng thể của cả nhà máy, vì như thế cũng sẽ phải tốn kém cho việc viết lại tất cả các chương trình.

- Giải pháp 5 là một giải pháp có tính chất thỏa hiệp. Hai máy tính cũ cùng hai hệ chương trình ĐH và PH được tận dụng triệt để (thỏa mãn điều ưu tiên mà cơ quan sử dụng đề xuất). Việc đưa thêm kho hàng dự trữ cùng với các cải tiến mới về khâu phát hiện địa chỉ phát hàng (nay xử lý trên hệ PH) và kiểm tra (nay xử lý trên hệ ĐH) đã đáp ứng được các mục tiêu, mà không gây tốn kém nhiều. Giải pháp 5 do đó đã được chọn.

4. Lập kế hoạch triển khai dự án

a) Hợp đồng triển khai dự án

Dự án xây dựng hệ thống mới đã được khẳng định. Cần có một hợp đồng giữa bên sử dụng hệ thống và bên xây dựng hệ thống để chốt những nội dung chủ yếu của dự án đó. Hợp đồng đó có thể là thành văn (nếu bên sử dụng và bên xây dựng thuộc hai cơ quan khác nhau) hoặc là không thành văn (nếu bên xây dựng lại là bộ phận của cơ quan chủ dự án) nhưng đều phải chứa đựng các nội dung chủ yếu sau:

- Vấn đề đặt ra và các nhu cầu về thông tin;
- Phạm vi và hạn chế;
- Mục tiêu và ưu tiên;
- Giải pháp và tính khả thi;
- Dự trù thiết bị và kinh phí;
- Phân công trách nhiệm và nhân sự;
- Phương pháp và tiến trình triển khai;

Bốn điểm đầu đã được xem xét ở trên, nay được đưa vào hợp đồng như là sự chấp thuận của cả đôi bên. Ba điểm sau sẽ được đề cập thêm ở dưới đây.

b) Dự trù thiết bị và kinh phí

- Dự trù thiết bị: Cần nhớ rằng ta đang ở giai đoạn khởi đầu dự án, chưa có đầy đủ các căn cứ để tính toán một cách chính xác cấu hình của thiết bị cần có cho hệ thống tương lai. Tuy nhiên việc dự trù thiết bị lại cần làm sớm, vì

việc đặt mua và lắp đặt thiết bị là cần có thời gian. Vả lại một số căn cứ như sau đã có thể ước lượng được:

- khối lượng dữ liệu cần lưu trữ lâu dài;
- các dạng làm việc (xử lý theo mẻ, từ xa, tương tác, trực tuyến v.v...);
- số lượng người dùng tối đa;
- khối lượng thông tin cần thu thập;
- khối lượng thông tin cần kết xuất, các tài liệu cần in ra

Trên các căn cứ này, ta đã có thể xác định được cấu hình của thiết bị, bao gồm:

- Mạng hay máy lẻ;
- Các thiết bị ngoại vi, đặc biệt là các thiết bị ngoại vi đặc dụng;
- Các đường truyền, các PLC (programmable logic controller);
- Các phần mềm cơ bản (hệ điều hành, hệ quản trị cơ sở dữ liệu, các ngôn ngữ lập trình v.v...);

Việc đặt mua các thiết bị cần được cân nhắc trên các mặt sau:

- Chất lượng và giá cả;
- Điều kiện giao hàng và lắp đặt;
- Huấn luyện;
- Bảo hành.

- Dự trù kinh phí: Ngoài kinh phí cho thiết bị và xây dựng địa điểm, còn phải có kinh phí cho quá trình triển khai dự án. Kinh phí này phải được thỏa thuận của đôi bên tham gia dự án, căn cứ trên các mặt:

- Khối lượng công việc và số người tham gia thực hiện qua các giai đoạn.
- Mức độ của các đòi hỏi về chất lượng sản phẩm, về thời hạn hoàn thành và về bảo hành sản phẩm.

Kinh phí cho dự án thường được phân bổ cho các giai đoạn triển khai dự án.

c) Tổ chức nhóm làm việc

Có hai cách tổ chức.

Cách tổ chức cổ điển có thể nói là ít nhiều còn mang sắc thái hành chính quan liêu. Một trưởng dự án được chỉ định. Người này phụ trách mọi thành viên

trong nhóm. Các thành viên này được phân loại thành người thiết kế, người phân tích, người lập trình v.v... với mức lương thấp dần. Công việc mỗi loại thành viên này được đóng kín. Đặc biệt, người phân tích thì từ chối lập trình vì cho như thế là dưới khả năng. Còn người lập trình thì không được đụng tới phân tích vì cho như thế là muốn vượt khung hay muốn lên lương. Đương nhiên sự thiếu hợp tác giữa các loại thành viên như thế chẳng mang lại lợi ích gì cho dự án.

Chính vì để chống lại quan điểm tổ chức cổ điển như trên, mà người ta đề nghị một cách tổ chức khác gọi là "Nhóm lập trình", một tên gọi không chính xác lắm, chỉ định một nhóm làm việc bình đẳng, trong đó có một trưởng nhóm, là một người có khả năng hơn mọi người và được mọi người tin tưởng. Cách tổ chức này do IBM đề xuất trong dự án "New-York Times" và được Tabourier, Rochfeld và Frank mô tả như sau: Nhóm lập trình được thành lập xung quanh một hạt nhân gồm có 3 người: một trưởng nhóm, một phó trưởng nhóm và một thư ký nhóm. Tư tưởng chủ đạo là lập một cấu trúc mềm dẻo, trong đó người trưởng và người phó có thể thay thế lẫn nhau và hai nhà chuyên môn này có thể trút mọi công việc hành chính cho người thư ký. Sự phân biệt có tính quan liêu giữa người thiết kế, người phân tích, người lập trình và những người khác trở nên mờ nhạt hoàn toàn: mọi người đều là lập trình viên và chẳng ai ngăn cấm mỗi người có thể có những kiến thức riêng biệt trên các lĩnh vực khác.

Lưu ý rằng tổ chức theo kiểu "Nhóm lập trình" hoàn toàn không có nghĩa là xem nhẹ các khâu phân tích và thiết kế, mà chỉ là để cho các phần việc phân tích, thiết kế và lập trình dễ hòa quyện với nhau hơn. Ở nước ta - may thay - phần lớn có nhóm triển khai dự án đều được tổ chức theo kiểu "Nhóm lập trình", ít thấy có sự phân công quá rạch ròi giữa phân tích, thiết kế và lập trình. Nhưng tiếc thay, nguồn cội của hình thức tổ chức này ở nước ta lại chính là ở sự xem nhẹ các việc phân tích và thiết kế. Không ít nơi người ta có thói quen "lập trình chay", nghĩa là lập trình mà không phân tích và thiết kế. "Lập trình chay" tạm thời có thể thực hiện trôi chảy ở các dự án nhỏ, nhưng không thể thực hiện mà không gây ra cảnh rối loạn ở các dự án lớn, với sự tham gia xây dựng của nhiều người (trên 10 người chẳng hạn).

d) Sự điều hành dự án

Vấn đề quyền điều hành dự án cũng nảy sinh khúc mắc. Trước đây, qua rất nhiều năm, không ai phản đối gì việc các nhà tin học nắm trọn quyền điều hành dự án. Nhưng rồi dần dần các người dùng có xu hướng tạo phản, đòi lại quyền điều hành dự án, vì nó liên quan quá trực tiếp đến họ.

Quả vậy, khi tầm cỡ của dự án còn bé - chẳng hạn thu hẹp trong nhiệm vụ lập một vài biểu mẫu thống kê cho một phòng bán hàng của xí nghiệp - thì người dùng còn ít bận tâm đến nó và có thể phó thác quyền điều hành dự án cho nhà tin học được. Nhưng khi dựa án là có tầm cỡ lớn, đụng đến các thông tin tổng thể, có ảnh hưởng sâu rộng đối với xí nghiệp, thì quyền điều hành tổng thể của dự án lại xứng đáng ở trong tay của người dùng. Cái khó là phải tìm cho được một người trong ban lãnh đạo xí nghiệp, vừa có khả năng về tin học, vừa nắm vững chiến lược phát triển của xí nghiệp để giữ quyền điều hành dự án. Nhân vật hiếm hoi này đôi khi cũng có thể tìm được. Nhưng để khỏi phải khó khăn tìm kiếm, trong trường hợp chung, ta lập một ban điều hành thay vì một người điều hành.

Ban điều hành gồm một số lượng hạn chế các thành viên, trong đó có cả nhà tin học và cả người dùng, được tổ chức bên trên nhóm làm việc của dự án và chịu trách nhiệm về việc đưa ra các quyết định có tính định hướng cho dự án, như là:

- quyết định các mục tiêu và chỉ ra cách đạt đến các mục tiêu đó;
- xác định các mức độ an toàn và các kiểm tra tương ứng;
- phân phối các nguồn lực xét thấy cần thiết về nhân lực, vật lực hay ngân quỹ;
- kiểm tra sự phát triển đúng đắn của dự án, đánh giá thường kỳ các kết quả đạt được so với các chỉ tiêu dự kiến;
- quyết định việc chọn lựa giải pháp khi nhóm làm việc phải đối mặt với nhiều phương án giải quyết có tác động sâu sắc tới sự hoạt động của xí nghiệp.

Ban điều hành làm việc theo kiểu nhóm họp đều kỳ, ở đó người trưởng nhóm làm việc báo cáo lại các việc đã làm, vạch ra các việc cần làm, các khoảng cách so với kế hoạch, các khó khăn vấp phải. Ban điều hành cân nhắc tình hình để đưa ra các quyết định quan trọng như là: cấp phát các phương tiện bổ sung, giảm mức độ cho một số mục tiêu, chấp nhận việc kéo dài một số kỳ hạn hay là một sự vượt trội ngân sách, và trong trường hợp nghiêm trọng là cho ngừng công việc.

đ) Tiến trình của dự án

Cần phải dự kiến tiến trình phát triển của dự án. Chẳng hạn sau đây là một tiến trình trong đó dự trù thời hạn và nhân lực cho 3 giai đoạn lớn phân tích, thiết kế và thực hiện:

	Người		Người/Tháng	Bắt đầu	Kết thúc
	CB tin học	Người dùng			
Phân tích	1	1	4	1/ 11/ 98	31/12/98
Thiết kế	3	1	12	15/ 1/ 99	15/4/99
Thực hiện	3	1	7,5	16/ 4/ 99	30/6/99
Tổng cộng số người/ tháng			23,5		

Tuy nhiên thường thì các giai đoạn lớn nói trên được phân chia thành các giai đoạn nhỏ hơn. Có nhiều cách phân chia.

Theo SIS (Swedish Standard Institution) thì có 9 giai đoạn

- Nghiên cứu tính khả thi;
- Xác định các mục tiêu;
- Nghiên cứu các thông tin;
- Nghiên cứu các xử lý;
- Nghiên cứu các hệ thống;
- Nghiên cứu chi tiết các hệ thống;
- Thiết kế các thành phần của các hệ thống;
- Cài đặt các hệ thống;
- Nghiên cứu sau cài đặt.

Phương pháp SDM (System Developent Methodology) lại đưa ra 7 bước:

- Định nghĩa vấn đề;
- Thiết kế sơ bộ;
- Thiết kế chi tiết;
- Phát triển các chương trình và các nhiệm vụ thủ công;
- Thử nghiệm;
- Chuyển đổi các thông tin và cài đặt hệ thống;
- Khai thác và bảo trì,

AFNOR (Association Française de Normalisation) thì cho rằng sự phân chia giai đoạn là tùy thuộc vào từng loại xí nghiệp, lấy thí dụ có thể là:

- Mở đầu;
- Nghiên cứu tính hợp thời;
- Thiết kế chi tiết;
- Thực hiện về phương diện tin học;
- Nghiên cứu về tổ chức;
- Cài đặt hệ thống;
- Đánh giá;
- Bảo trì.

Sự khác biệt giữa các phương pháp trên (về cách phân chia giai đoạn và về từ ngữ sử dụng) thực ra là không cơ bản.

Các phương pháp mà ta xem là cổ điển này có chung một bản chất là chấp nhận một tiến trình tuyệt đối tuyến tính về sự tiếp diễn các giai đoạn. Trong tiến trình tuyến tính đó, thì giai đoạn này có hoàn thành mới được phép chuyển sang giai đoạn khác. Các giai đoạn không được chờm lên nhau. Cách làm này luôn luôn dẫn tới nguy cơ kéo dài thời hạn mỗi giai đoạn và rốt cục là kéo dài thời hạn hoàn thành của toàn bộ dự án.

Để khắc phục nhược điểm này, có những phương pháp cho phép các giai đoạn chờm lên nhau, nghĩa là cho phép bắt đầu một giai đoạn, một công việc khi chưa thật sự đầy đủ các thông tin, và dù rằng sau này sẽ phải quay lại bổ sung các thông tin đó.

Phương pháp phân tích trên xuống, được quán triệt trong các chương sau của tập sách này, cũng có thể dẫn tới một tiến trình phi tuyến tính. Đó là tiến trình lặp và tăng trưởng dần dần (xem lại §3.1 ở Chương I).

Ở giai đoạn xác lập dự án này, vấn đề chỉ là chọn một phương pháp và một tiến trình triển khai cho thích hợp với hệ thống cần xây dựng, cũng như thích hợp với kinh nghiệm, thói quen và cả với công cụ sẵn có của nhóm thực hiện dự án. Trên cơ sở đó mà xác định một lịch biểu cho dự án.

Chương III

PHÂN TÍCH HỆ THỐNG VỀ CHỨC NĂNG

Trong chương này ta xét sự phân tích hệ thống về mặt chức năng, mà mục đích là lập một mô hình chức năng của hệ thống, nhằm trả lời câu hỏi "Hệ thống làm gì?"

Đầu tiên ta để cập một số mô hình và phương tiện được sử dụng để diễn tả các chức năng. Sau đó ta sẽ đề cập cách tiến hành phân tích hệ thống về chức năng, tức là sẽ nói rõ làm thế nào để có thể đi sâu vào bản chất và đi sâu vào chi tiết của hệ thống về mặt chức năng. Phương pháp được chọn lựa để trình bày ở đây là phương pháp SA (Structured Analysis) do De Marco và những người khác đề xuất.

§1. CÁC MÔ HÌNH VÀ PHƯƠNG TIỆN DIỄN TẢ CHỨC NĂNG

1. Các mức độ diễn tả chức năng

Nhắc lại rằng chức năng nói ở đây là chức năng xử lý thông tin, vì hệ thống mà ta đề cập là hệ thống xử lý thông tin hoặc hệ thống điều khiển quá trình, chứ không phải là hệ thống xử lý vật chất hay năng lượng. Ba từ *chức năng, xử lý* và *quá trình* ở đây được coi là đồng nghĩa.

Việc diễn tả chức năng, tùy theo hoàn cảnh và yêu cầu, mà có thể thực hiện ở những mức độ khác nhau: ,

a) Diễn tả vật lý và diễn tả lôgic

Sự diễn tả chức năng ở *mức độ vật lý* đòi hỏi phải nói rõ cả mục đích và cách thực hiện của quá trình xử lý, nói cách khác là phải trả lời cả 2 câu hỏi: "Làm gì? và "Làm như thế nào?". Câu hỏi "Làm như thế nào?" thể hiện ở các khía cạnh như "Dùng phương pháp gì? Biện pháp gì?", "Dùng công cụ gì? (Tự động hay thủ công), "Ai làm?", "Ở đâu?", "Lúc nào?"...

Sự diễn tả chức năng ở *mức độ lôgic*, trái lại, chỉ tập trung trả lời câu hỏi "Làm gì?" mà gạt bỏ câu hỏi "Làm như thế nào?", nghĩa là chỉ diễn tả mục

đích, bản chất của quá trình xử lý, mà bỏ qua các yếu tố về thực hiện, về cài đặt (thường gọi là các yếu tố vật lý) như phương pháp, phương tiện, tác nhân, địa điểm, thời gian...

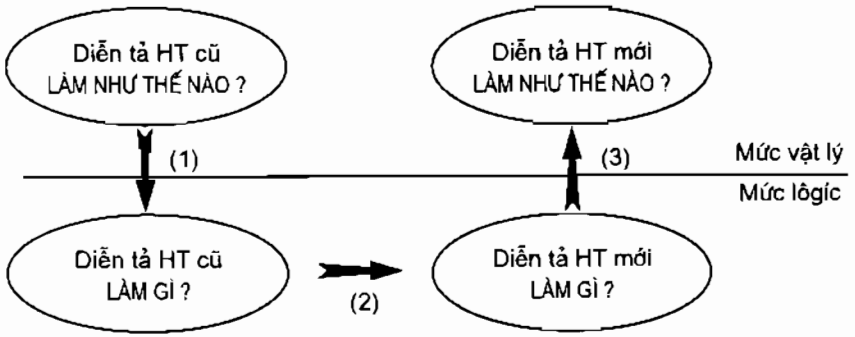
Trong giai đoạn khảo sát sơ bộ một hệ thống có sẵn, ta phải ghi nhận nguyên xi những gì đang diễn ra trong thực tế. Vậy lúc đó ta phải dùng sự diễn tả ở mức độ vật lý.

Tuy nhiên các yếu tố vật lý thường làm che khuất bản chất của hệ thống, làm lu mờ hay biện minh cho các bất hợp lý của hệ thống (làm ta tưởng là hợp lý). Ta thường nghe các vị giám đốc giải bày: "Tôi cho làm thế này, thế nọ là do tôi thiếu người, thiếu phương tiện, do địa điểm phân tán, thời hạn gấp gấp v.v...". Để phanh phui bản chất, nói rõ sự thật bất hợp lý, ở giai đoạn phân tích hệ thống ta phải loại bỏ mọi yếu tố vật lý, và diễn tả các chức năng của hệ thống ở mức độ lôgic. Đối với hệ thống mới (hệ thống cần xây dựng), thì một sự mô tả lôgic một cách hoàn chỉnh và hợp lý là cần thiết trước khi tính đến các biện pháp về cài đặt.

Giai đoạn thiết kế là lúc tính đến các biện pháp cài đặt đó.

Vậy sang giai đoạn thiết kế, ta lại phải diễn tả sự hoạt động của hệ thống ở mức độ vật lý, với đầy đủ các yếu tố về cài đặt và thực hiện.

Có thể tóm tắt sự thay đổi mức độ diễn tả vật lý/ lôgic trong hình vẽ sau, trong đó các bước chuyển đổi (1) và (2) thuộc giai đoạn phân tích, bước chuyển đổi (3) thuộc giai đoạn thiết kế.



Hình III.1. Một trình tự mô hình hóa HT

b) Diễn trả đại thể và diễn tả chi tiết

Ở mức độ đại thể, thì một chức năng được mô tả dưới dạng "hộp đen". Nội dung bên trong hộp đen không được chỉ rõ, nhưng các thông tin vào và ra hộp đen thì lại được chỉ rõ (Hình II.2).

Hình III.2. Một hộp đen

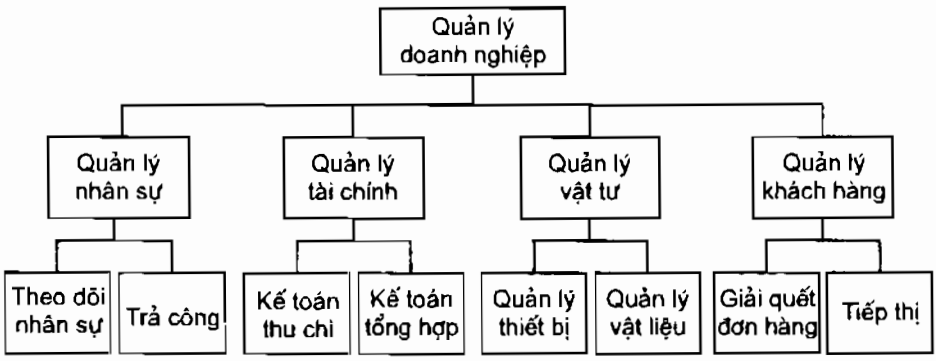
Ở mức độ chi tiết thì nội dung của quá trình xử lý phải được chỉ rõ hơn. Thông thường thì cần chỉ ra các chức năng con và mối liên hệ về thông tin và điều khiển giữa những chức năng con đó.

Vì các chức năng con thường vẫn còn phức tạp, nên lại phải diễn tả chúng một cách chi tiết hơn, thông qua các chức năng nhỏ hơn. Cứ thế tiếp tục, ta sẽ có một sự phân cấp trong mô tả. Ở mức cuối cùng, thì các chức năng là khá đơn giản. Bấy giờ ta có thể diễn tả trực tiếp quá trình xử lý của nó, mà không cần vỡ nó thành chức năng con nữa. Sự mô tả trực tiếp một chức năng được gọi là sự *đặc tả*.

Sự mô tả đại thể, chi tiết hay đặc tả cũng như sự mô tả vật lý hay lôgic được sử dụng tùy lúc, tùy nơi trong phân tích và thiết kế hệ thống. Dưới đây là một số mô hình và phương tiện diễn tả chức năng được vận dụng ở những mức độ mô tả khác nhau đó.

2. Các biểu đồ phân cấp chức năng

Biểu đồ phân cấp chức năng (BPC) là một loại biểu đồ diễn tả sự phân rã dần dần các chức năng từ đại thể đến chi tiết. Mỗi nút trong biểu đồ là một chức năng, và quan hệ duy nhất giữa các chức năng, diễn tả bởi các cung nối liền các nút, là quan hệ bao hàm. Như vậy BPC tạo thành một cấu trúc cây (Hình III.3).



Hình III.3. Một BPC

Đặc điểm của các BPC là:

- Cho một cách nhìn khái quát, dễ hiểu, từ đại thể đến chi tiết về các chức năng, nhiệm vụ cần thực hiện (thường ở mức diễn tả lôgic).

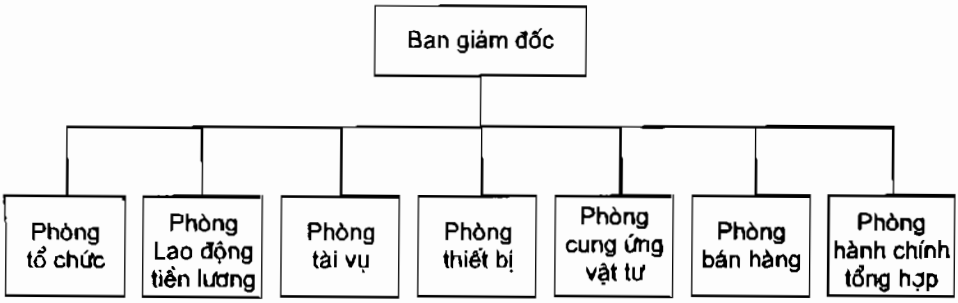
- Rất dễ thành lập, bằng cách phân rã dần dần các chức năng từ trên xuống.

- Có tính chất tĩnh, bởi chúng chỉ cho thấy các chức năng mà không cho thấy trình tự xử lý.

- Thiếu vắng sự trao đổi thông tin giữa các chức năng.

Vì những đặc điểm kể trên mà BPC thường được sử dụng làm mô hình chức năng trong bước đầu phân tích, hoặc cho các hệ thống đơn giản. Nếu hệ thống là phức tạp thì một mô hình chức năng dưới dạng BPC là quá sơ lược và các thiếu sót nêu trong hai đặc điểm cuối ở trên là không thể châm chước được. Lúc đó ta thường dùng các biểu đồ luồng dữ liệu (nói ở dưới) thay cho BPC.

Chú ý: Cần phân biệt BPC với sơ đồ tổ chức của một cơ quan. Sơ đồ tổ chức thể hiện các bộ phận, các tổ chức hợp thành cơ quan. Bởi sự phân cấp quản lý thường được áp dụng trong các cơ quan, cho nên sơ đồ tổ chức cũng thường có dạng cây. Nói chung là có sự tương ứng giữa tổ chức và chức năng. Tuy nhiên sự tương ứng đó không nhất thiết là tương ứng 1-1. Vì vậy mà BPC và sơ đồ tổ chức của cùng một cơ quan có những khác biệt; không những có khác biệt về tên của các nút trên biểu đồ (một đằng là tên chức năng, một đằng là tên bộ phận) mà còn có khác biệt về cấu trúc cây của chúng. Chẳng hạn với doanh nghiệp đề cập trong Hình III.3 thì sơ đồ tổ chức lại có thể như ở Hình III.4.



Hình III.4 Một sơ đồ tổ chức

3. Các lưu đồ hệ thống

Lưu đồ hệ thống (LH) là một loại biểu đồ nhằm diễn tả quá trình xử lý thông tin của một hệ thống với các yêu cầu sau:

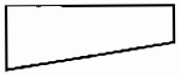
- sự diễn tả là ở mức vật lý;

- chỉ rõ các công việc (chức năng xử lý) phải thực hiện;

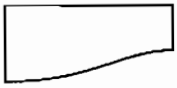
- chỉ rõ trình tự các công việc và các thông tin được chuyển giao giữa các công việc đó.

Hình III.5 cho một lưu đồ hệ thống diễn tả một quá trình giải quyết một đơn đặt hàng. Lưu đồ này được vẽ trên một không gian hai chiều: chiều ngang chỉ các địa điểm, chiều dọc (từ trên xuống) chỉ trình tự thời gian. Các nút trong lưu đồ được vẽ theo các quy ước do hãng IBM đề xuất như sau:

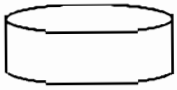
chức năng xử lý thông tin



chứng từ (trên giấy)



danh sách (in trên giấy)

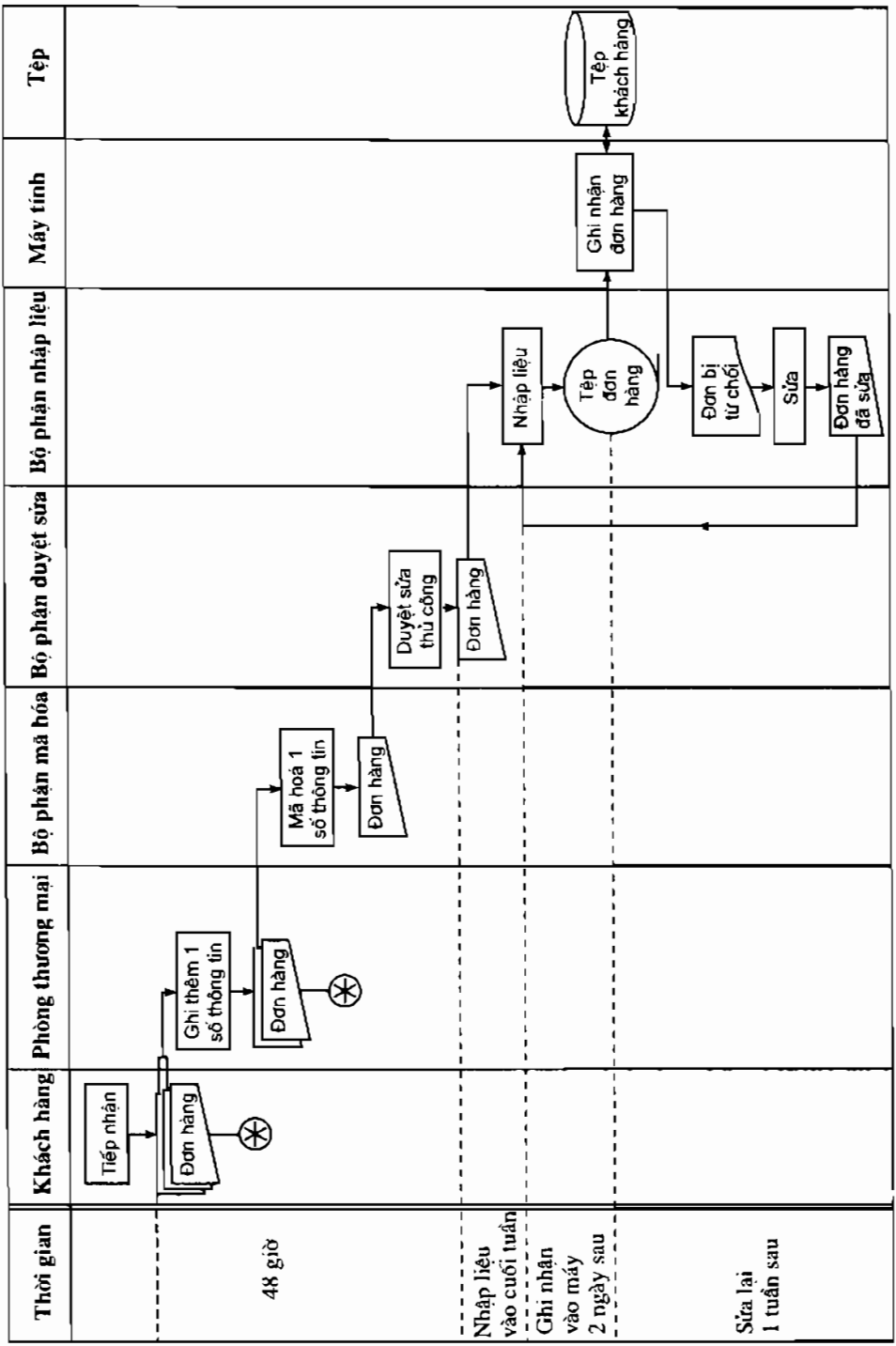


tệp trên đĩa từ

tệp trên băng từ

lưu (tại chỗ)

LH phục vụ cho sự diễn tả ở mức độ vật lý và cũng khá dễ hiểu. Chúng thường được sử dụng ở giai đoạn khảo sát hệ thống cũ, hoặc ở giai đoạn thiết kế. Tuy nhiên chúng chỉ thịnh hành vào các năm 60 - 70, còn ngày nay chúng ít được sử dụng.



Hình III.5 Một lưu đồ hệ thống

4. Biểu đồ luồng dữ liệu

Biểu đồ luồng dữ liệu (BLD) là một loại biểu đồ nhằm mục đích diễn tả một quá trình xử lý thông tin với các yêu cầu sau:

- sự diễn tả là ở mức lôgic, nghĩa là nhằm trả lời câu hỏi "Làm gì?", mà bỏ qua câu hỏi "Làm như thế nào?".

- chỉ rõ các chức năng (con) phải thực hiện để hoàn tất quá trình xử lý cần mô tả.

- chỉ rõ các thông tin được chuyển giao giữa các chức năng đó, và qua đó phần nào thấy được trình tự thực hiện của chúng.

Vì yêu cầu diễn tả ở mức lôgic, cho nên so với lưu đồ hệ thống, thì BLD thiếu hẳn các khả năng biểu diễn về cách làm (chẳng hạn xử lý thủ công hay bằng máy tính), về các giá mang thông tin (chẳng hạn giấy, băng từ đĩa từ), về địa điểm, thời gian, tác nhân xử lý. Các BLD chỉ được phép sử dụng năm loại *yếu tố biểu diễn* sau đây:

(1) *Các chức năng:*

- *Định nghĩa*: Một chức năng là một quá trình biến đổi dữ liệu (thay đổi giá trị, cấu trúc, vị trí của một dữ liệu, hoặc từ một số dữ liệu đã cho, tạo ra một dữ liệu mới).
- *Biểu diễn*: Một chức năng được biểu diễn (trong BLD) bởi một hình tròn hay một hình ôvan (thường được gọi là một bong bóng), bên trong có tên của chức năng đó.

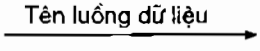
Tên chức năng phải là một động từ, có thêm bổ ngữ nếu cần, cho phép hiểu một cách vắn tắt chức năng làm gì. Chẳng hạn:

(2) *Các luồng dữ liệu:*

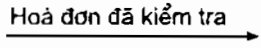
- *Định nghĩa*: Một luồng dữ liệu là một tuyến truyền dẫn thông tin vào hay ra một chức năng nào đó.

Khi nói tuyến truyền dẫn thông tin thì ta hiểu là ở đây có một thông tin được chuyển đến một chức năng để được xử lý, hoặc chuyển đi khỏi một chức năng như một kết quả xử lý, bất kể hình thức truyền dẫn là gì (bằng tay, qua máy tính, bằng fax hay điện thoại v.v...). Thông tin ở đây có thể là một dữ liệu đơn (chẳng hạn: tên khách hàng), cũng có thể là một dữ liệu có cấu trúc (chẳng hạn: hóa đơn). Lại chú ý rằng mọi luồng dữ liệu là phải vào hay ra một chức năng nào đó, vậy trong hai đầu của một luồng dữ liệu (đầu đi và đầu đến), ít nhất phải có một đầu dính tới một chức năng.

- *Biểu diễn*: Một luồng dữ liệu được vẽ trong một BLD dưới dạng một mũi tên, trên đó có viết tên của luồng dữ liệu.



Tên luồng dữ liệu phải là một danh từ, kèm thêm tính ngữ nếu cần, cho phép hiểu vắn tắt nội dung của dữ liệu được chuyển giao. Chẳng hạn:

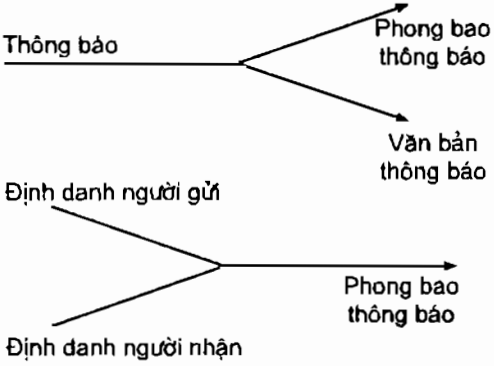


Trong trường hợp dữ liệu được chuyển giao là một dữ liệu có cấu trúc, thì khi cần ta có thể vẽ luồng dữ liệu rẽ nhánh (thành các luồng thành phần), hoặc chập lại (từ các luồng thành phần). Chẳng hạn, trong viễn thông, người ta dùng khái niệm "thông báo", với định nghĩa sau:

Thông báo = Phong bao thông báo + Văn bản thông báo

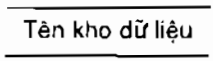
Phong bao thông báo = Định danh người gửi + Định danh người nhận

Vậy có thể vẽ (khi cần) các luồng như sau:

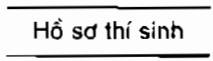


(3) *Các kho dữ liệu*

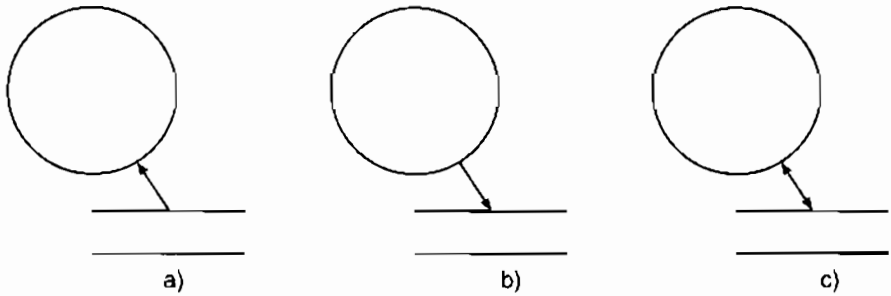
- *Định nghĩa*: Một kho dữ liệu là một dữ liệu (đơn hay có cấu trúc) được lưu lại, để có thể được truy nhập nhiều lần về sau.
- *Biểu diễn*: Một kho dữ liệu được vẽ trong một BLD dưới dạng hai đoạn thẳng nằm ngang, kẹp giữa tên của kho dữ liệu.



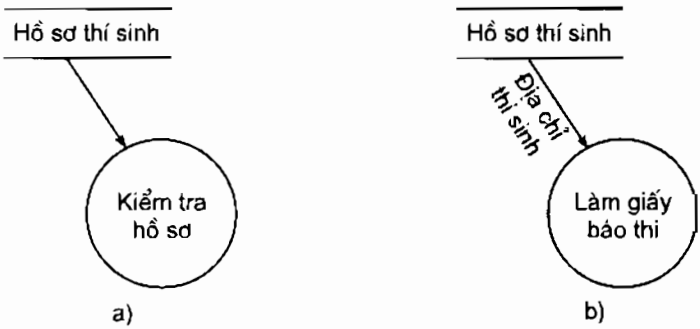
Tên của kho dữ liệu phải là một danh từ, kèm theo tính ngữ nếu cần, cho phép hiểu một cách vắn tắt nội dung của dữ liệu được lưu giữ. Chẳng hạn:



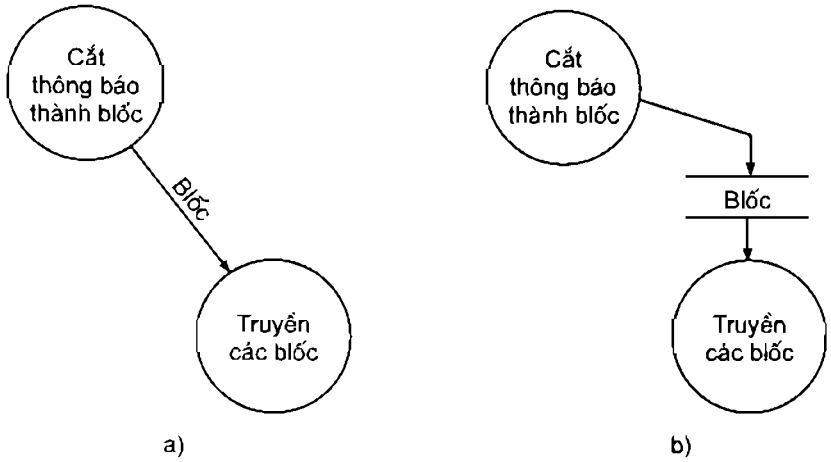
Việc sử dụng các kho dữ liệu trong BLD tuân thủ một số quy tắc diễn tả và hiểu nghĩa thể hiện như trong các Hình III.6, III.7, III.8, III.9, III.10 sau đây:



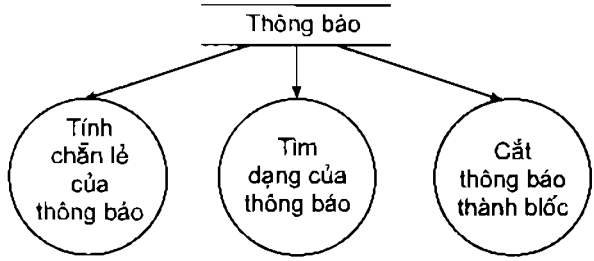
Hình III.6 Quy tắc sử dụng kho dữ liệu:
Ba cách truy nhập: a) Đọc; b) Viết; c) Đọc và viết



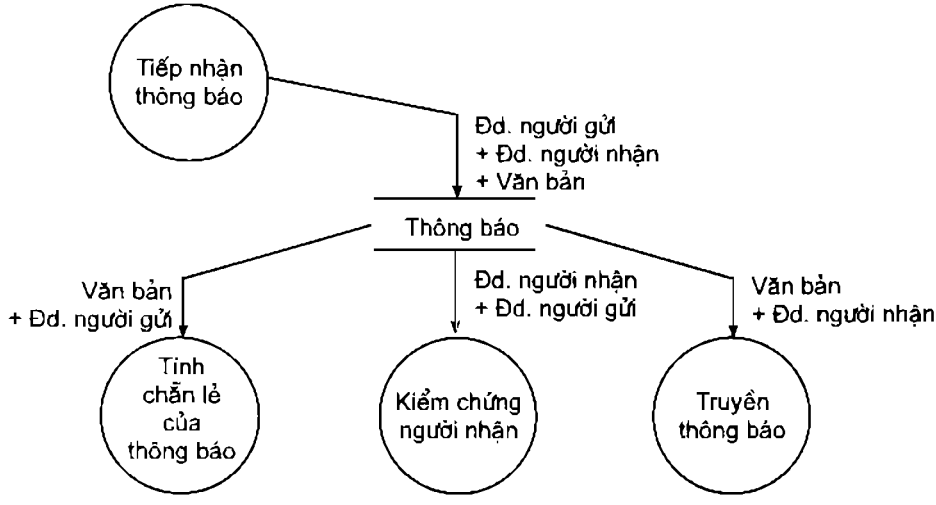
Hình III. 7 Quy tắc sử dụng kho dữ liệu:
(a) Truy nhập toàn bộ dữ liệu: Luồng dữ liệu không cần mang tên.
(b) Truy nhập một phần dữ liệu: Luồng dữ liệu phải mang tên chỉ rõ thành phần truy nhập.



Hình III. 8 Quy tắc sử dụng kho dữ liệu:
(a) Không có kho: Thông tin được xử lý ngay
(b) Có kho: Thông tin được xử lý ở thời điểm khác với thời điểm được sinh ra.



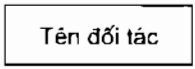
Hình III. 9 Quy tắc sử dụng kho dữ liệu: Truy nhập nhiều lần cùng một thông tin



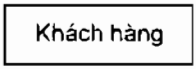
Hình III. 10 Quy tắc sử dụng kho dữ liệu:
Sử dụng các thông tin với một trật tự khác với trật tự khi sản sinh

(4) *Các đối tác*

- *Định nghĩa*: Một *đối tác* (còn gọi là *tác nhân ngoài*, hay *điểm mút*) là một thực thể ngoài hệ thống, có trao đổi thông tin với hệ thống.
- *Biểu diễn*: Đối tác trong BLD được vẽ bằng một hình chữ nhật, bên trong có tên đối tác.



Tên đối tác phải là một danh từ, cho phép hiểu vắn tắt đối tác là ai, hoặc là gì (người, tổ chức, thiết bị, tệp v.v...). Chẳng hạn:



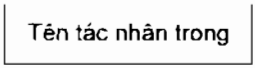
Đối tác chỉ phát (mà không nhận) thông tin đến hệ thống thường được gọi là *nguồn*. Còn đối tác chỉ nhận (mà không phát) thông tin từ hệ thống thường được gọi là *vực*.

(5) *Các tác nhân trong*

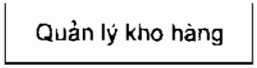
- *Định nghĩa*: Một *tác nhân trong* là một chức năng hay một hệ con của hệ thống, được mô tả ở một trang khác của mô hình, nhưng có trao đổi thông tin với các phần tử thuộc trang hiện tại của mô hình.

Như vậy tác nhân trong xuất hiện trong BLD chỉ để làm nhiệm vụ tham chiếu.

- *Biểu diễn*: Tác nhân trong trong BLD được vẽ dưới dạng một hình chữ nhật thiếu cạnh trên, trong đó viết tên tác nhân trong (chức năng hay hệ thống con).



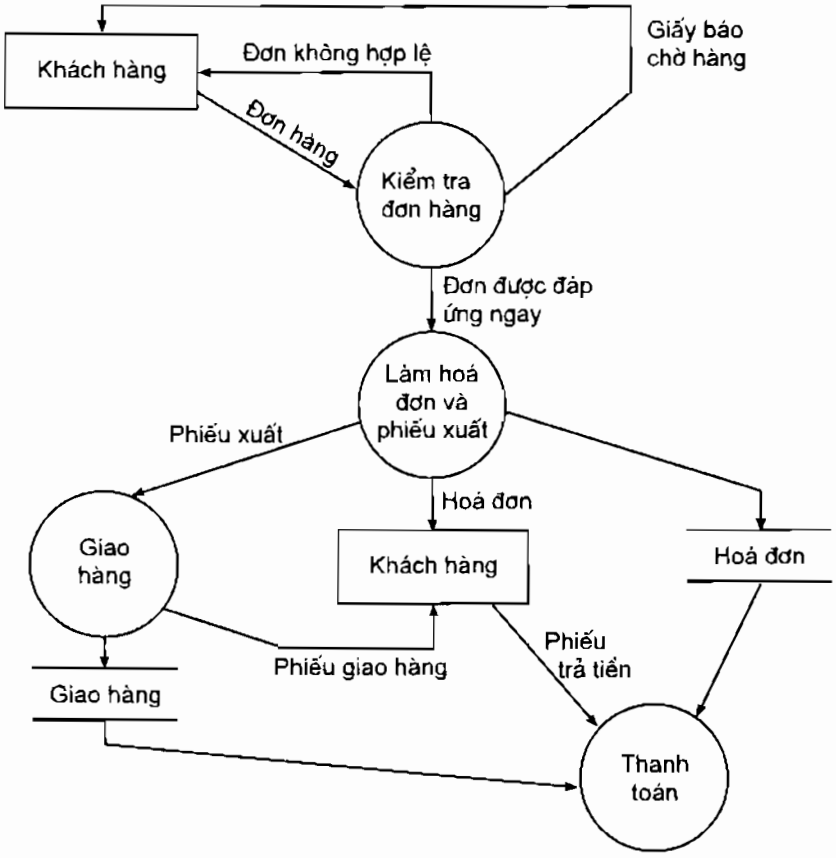
Tên tác nhân trong phải là một động từ, kèm thêm bổ ngữ khi cần. Chẳng hạn:



Trên đây là năm yếu tố biểu diễn được phép sử dụng trong một BLD. Với sự hạn chế khá ngặt nghèo đó thì các BLD ít có khả năng tiếp nhận các yếu tố vật lý, và giữ được sự diễn tả ở mức lôgic. Tuy nhiên, như sau này ta sẽ thấy,

nếu người phân tích sơ ý, thì vẫn có khả năng để lọt lưới các yếu tố vật lý xâm nhập vào BLD. Vì vậy mà vẫn còn phải có một nhiệm vụ cần làm trong quá trình phân tích hệ thống là gạt bỏ các yếu tố vật lý ra khỏi BLD.

Thí dụ: BLD diễn tả một quy trình bán hàng theo đơn đặt hàng (Hình III.11)



Hình III.11 Một BLD

Chú thích:

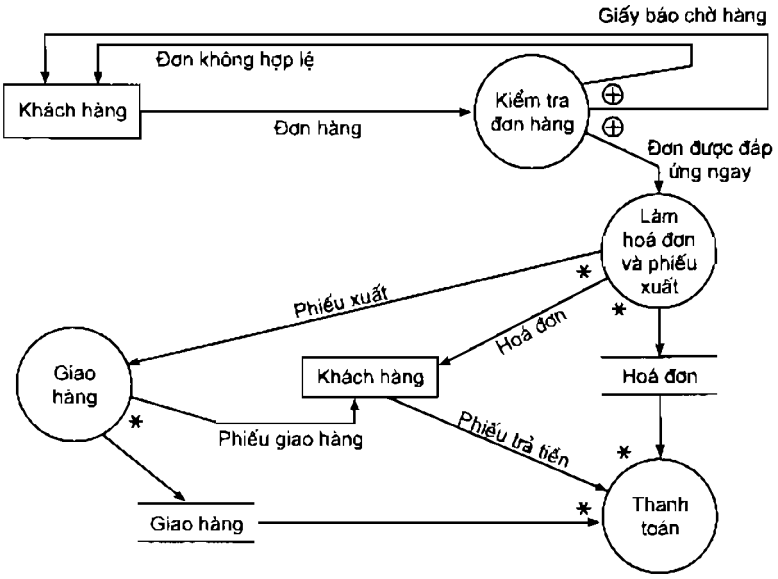
1/ Trong BLD, một đối tác (như đối tác khách hàng ở trên) cũng như một kho dữ liệu hay một tác nhân trong có thể được vẽ lặp lại, mà vẫn được hiểu như là một, chỉ vì lý do trình bày nhằm tránh sự chồng chéo của các luồng dữ liệu. Tuy nhiên các chức năng và các luồng dữ liệu thì không được vẽ lặp lại.

2/ Một nhận xét rất quan trọng là: Trong các BLD thì mọi thông tin xuất hiện (tức là các luồng dữ liệu và các kho dữ liệu) đều là những thông tin thuộc lĩnh vực ứng dụng. Chẳng hạn trong BLD ở Hình III.11, thì đó là những thông tin thuộc lĩnh vực bán hàng. Các chức năng cũng đều là các chức năng xử lý loại thông tin đó. Tuyệt nhiên không có mặt các thông tin điều khiển, nghĩa là những thông tin không phải để phản ánh lĩnh vực ứng dụng, mà là để tham gia vào các quyết định điều khiển sự thực hiện các chức năng xử lý (tức là để khởi động, ngưng ngắt, đồng bộ hóa, cho thực hiện đồng thời các chức năng xử lý). Vậy trong BLD, các chức năng được khởi động và thực hiện ra sao? Có một nguyên tắc ngầm định giải quyết việc này, ấy là nguyên tắc *kích hoạt bằng dữ liệu* (data - triggered) mà nội dung là:

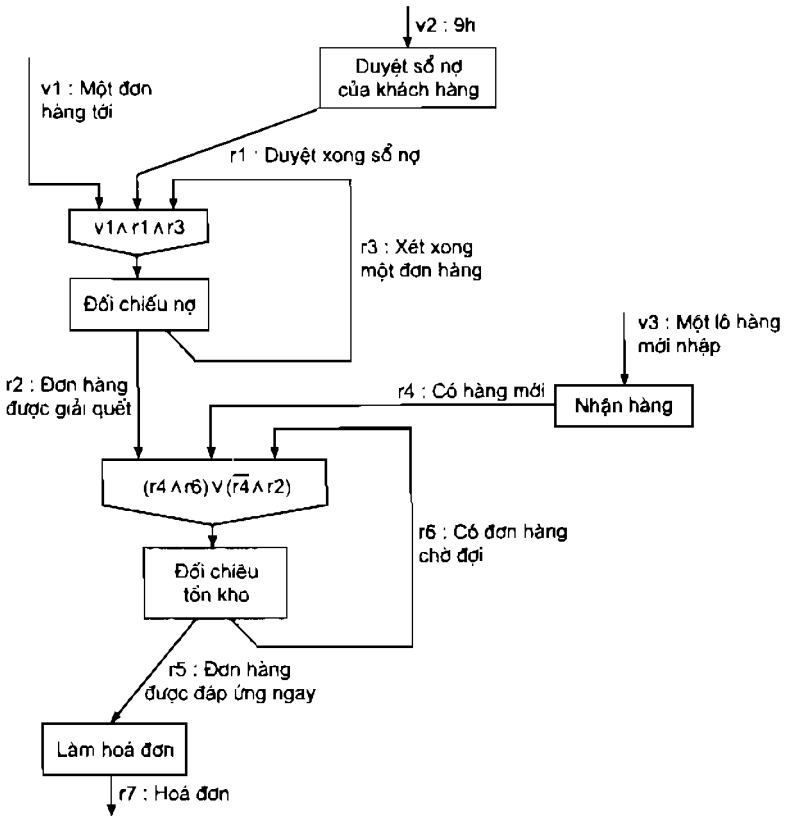
- Một chức năng trong BLD được khởi động khi nó hội đủ các dữ liệu cần thiết (ở đầu vào).
- Khi được khởi động, thì một chức năng được xem là được thực hiện vô cùng nhanh, nghĩa là các dữ liệu ở đầu ra có ngay lập tức.

Như vậy, trong BLD các yếu tố thời gian và điều khiển bị bỏ qua. Vậy nó có sự hạn chế đáng kể trong khả năng biểu diễn. Chính vì sự hạn chế đó, mà sau này, khi xét tới động thái của hệ thống (Chương VI), các yếu tố về điều khiển và biến đổi hành vi trong thời gian (rất quan trọng trong các hệ thống thời gian thực) sẽ được nghiên cứu, và bấy giờ ta phải sử dụng thêm một loại biểu đồ nữa, gọi là Biểu đồ luồng điều khiển để phối hợp trong sự diễn tả hệ thống với các BLD. Tuy nhiên lúc đó, thì nguyên tắc kích hoạt bằng dữ liệu nói trên vẫn là quan trọng, bởi vì ở đâu mà sự diễn tả về điều khiển không thể hiện tường minh, thì ở đó nguyên tắc ngầm định kích hoạt bằng dữ liệu vẫn được áp dụng.

3/ Một số phương pháp phân tích sử dụng một số loại BLD trong đó có đưa thêm ít nhiều khả năng diễn tả sự điều khiển. Dưới đây là hai thí dụ: Hình III. 12 cho lại BLD diễn tả quy trình bán hàng như ở Hình III.11, song có thêm các ký hiệu xác định các luồng vào/ ra một chức năng là có loại trừ lẫn nhau hay không, với quy ước: *: VÀ, ⊕: HOẶC CÓ LOẠI TRỪ, ○: HOẶC KHÔNG LOẠI TRỪ. Hình III. 13 cho một BLD dùng trong phương pháp MERISE, ở đó mỗi chức năng (diễn tả bởi hình chữ nhật) có gắn thêm một điều kiện đồng bộ hóa ở đầu vào. Chức năng chỉ được thực hiện khi điều kiện này đúng (quy ước: ∧: VÀ, ∨ : HOẶC, —: KHÔNG)



Hình III. 12. Một BLD mở rộng



Hình III. 13 Một biểu đồ MERISE

5. Các phương tiện đặc tả chức năng

a) Đặc tả chức năng

Một điểm chung trong việc sử dụng BPC và BLD là để diễn tả một chức năng phức tạp, ta phân rã nó ra thành nhiều chức năng con đơn giản hơn. Nói cách khác là từ một "hộp đen", ta giải thích nó bằng cách tách nó ra thành nhiều "hộp đen". Có vẻ như đó là một sự luẩn quẩn, song thực ra là đã có sự tiến bộ, vì các chức năng con thu được là đơn giản hơn trước. Muốn đẩy tới sự tiến bộ đó, ta lại tiếp tục phân rã các chức năng con này. Sự lặp lại quá trình phân rã (thông qua các BPC hay BLD) đương nhiên tới một lúc phải dừng lại. Các chức năng thu được ở mức cuối cùng, đã là rất đơn giản, cũng vẫn cần được giải thích (nếu không thì vẫn cứ là "hộp đen"). Bấy giờ sự giải thích chức năng phải được thực hiện bởi những phương tiện diễn tả trực tiếp (khác với các BPC và BLD). Gọi đó là sự *đặc tả* chức năng, thường gọi tắt là *P-Spec* (Process Specification).

Một đặc tả chức năng thường được trình bày một cách ngắn gọn, không vượt quá một trang A4, và gồm hai phần:

- Phần đầu đề gồm:

- Tên chức năng,

- Các dữ liệu vào,

- Các dữ liệu ra.

- Phần thân mô tả nội dung xử lý, ở đó thường sử dụng các phương tiện mô tả sau đây (liệt kê theo trật tự ưu tiên giảm dần):

- Các phương trình toán học.

- Các bảng quyết định hay cây quyết định.

- Các sơ đồ khối.

- Các ngôn ngữ tự nhiên cấu trúc hóa.

(Không nên sử dụng ngôn ngữ tự do).

Hình III. 14 cho một thí dụ đặc tả chức năng, trong đó phần thân dùng một phương trình toán học làm phương tiện mô tả. Các phương tiện còn lại (trong danh sách trên) sẽ được đề cập kỹ hơn một chút ở các mục nhỏ tiếp sau:

Đầu đề:

Tên chức năng: Tính kết quả bảo vệ đồ án

Đầu vào: Điểm của người phản biện,
Điểm của người hướng dẫn,
Số các ủy viên hội đồng
Điểm của từng ủy viên hội đồng

Đầu ra: Kết quả bảo vệ

Thân

Kết quả bảo vệ = (Điểm của người phản biện
+ Điểm của người hướng dẫn
$+ \frac{\sum \text{Điểm của các ủy viên hội đồng}}{\text{Số các ủy viên hội đồng}}) / 3$

Hình III. 14 Một P - Spec

b) Các bảng quyết định và cây quyết định

Chúng được sử dụng khi chức năng được đặc tả thực chất là một sự phân chia các trường hợp tùy thuộc một số điều kiện vào. Ứng với mỗi trường hợp thì có một sự chọn lựa khác biệt một số hành động (hay giá trị) ra nào đó.

Số các giá trị có thể của mỗi điều kiện vào phải là hữu hạn. Chẳng hạn:

"Là thương binh" có thể lấy giá trị Đúng (Đ) hay Sai (S).

"Điều kiện tuổi tác" có thể lấy 4 giá trị:

+ Tuổi thơ (dưới 13 tuổi)
+ Tuổi trẻ (từ 13 đến 29 tuổi)
+ Trung niên (từ 30 đến 59 tuổi)
+ Tuổi già (từ 60 trở lên)

Như vậy số các trường hợp có thể có là được biết trước (bằng tích của các số những giá trị có thể của các điều kiện vào). Nhờ vậy ta không để sót các trường hợp. Đó là một ưu điểm đáng kể của các bảng quyết định và các cây quyết định.

Bảng quyết định là một bảng hai chiều, trong đó một chiều (có thể là chiều ngang hay chiều dọc) được tách làm hai phần: một phần cho các điều kiện vào

và phần kia cho các hành động hay các biến ra. Chiều thứ hai là các trường hợp có thể xảy ra tùy thuộc giá trị của các điều kiện. Ứng với mỗi trường hợp (là cột hay là dòng), thì các hành động chọn lựa sẽ được đánh dấu ×, hoặc nếu cái ra là các biến, thì cho các giá trị tương ứng của các biến đó.

		Các trường hợp							
Các điều kiện	Điều kiện X	Đ	Đ	Đ	Đ	S	S	S	S
	Điều kiện Y	Đ	Đ	S	S	Đ	Đ	S	S
	Điều kiện Z	Đ	S	Đ	S	Đ	S	Đ	S
Các hành động	Hành đồng A							×	×
	Hành đồng B						×		
	Hành đồng C	×	×	×	×	×			
	Hành đồng D	×	×	×	×	×		×	×

Hình III. 15 Một bảng quyết định

Thí dụ: Một cửa hàng quy định:

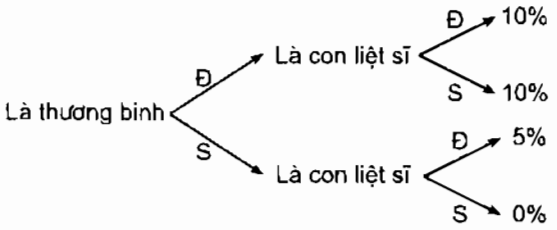
- Giảm giá 10% cho thương binh.
- Giảm giá 5% cho con liệt sĩ.
- Không được phép hưởng hai tiêu chuẩn (bấy giờ lấy mức cao nhất)

Như vậy chức năng "Xác định mức giảm giá cho khách hàng" được đặc tả bằng bảng quyết định như trong Hình III.16.

Là thương binh	Đ	Đ	S	S
Là con liệt sĩ	Đ	S	Đ	S
Giảm giá 10%	×	×		
Giảm giá 5%			×	
Giảm giá 0%				×

Hình III.16 Bảng quyết định "Giảm giá"

Cây quyết định chỉ là một biến tướng của bảng quyết định; nó phân chia các trường hợp nhờ cấu trúc cây, thay vì cấu trúc bảng. Chẳng hạn tương ứng bảng quyết định ở Hình III.16 ta có cây quyết định trong Hình III.17.



Hình III.17 Cây quyết định "giảm giá"

Chú thích: Có thể đơn giản hóa các bảng quyết định bằng cách gộp từng cặp trường hợp thỏa mãn hai điều:

- giống nhau hoàn toàn trên phần hành động,
- chỉ khác nhau một dòng trên phần điều kiện.

Như vậy bảng quyết định ở Hình III.15 sẽ trở thành bảng quyết định như ở Hình II.18.

Điều kiện X	Đ	S	S	S
Điều kiện Y	–	Đ	Đ	S
Điều kiện Z	–	Đ	S	–
Hành đồng A				×
Hành đồng B			×	
Hành đồng C	×	×		
Hành đồng D	×	×		×

Hình III. 18 Bảng quyết định đơn giản hóa

c) Các sơ đồ khối

Sơ đồ khối là loại biểu đồ diễn tả giải thuật quen thuộc và ưa dùng với các người mới học lập trình, vì nó đơn giản, dễ hiểu. Với lập trình nâng cao, thì nó bộc lộ nhiều nhược điểm, cho nên nó lại ít được ưa dùng: nó khuyến khích việc

sử dụng tràn lan GOTO, nó không thể hiện rõ ba cấu trúc điều khiển cơ bản (tuần tự, chọn, lặp), nó hỗ trợ kém cho lập trình trên xuống và càng tỏ ra gượng ép với lập trình đệ quy v.v... Tuy nhiên với nhiệm vụ đặc tả các chức năng đơn giản mà ta cần ở đây, thì nó đáp ứng được yêu cầu.

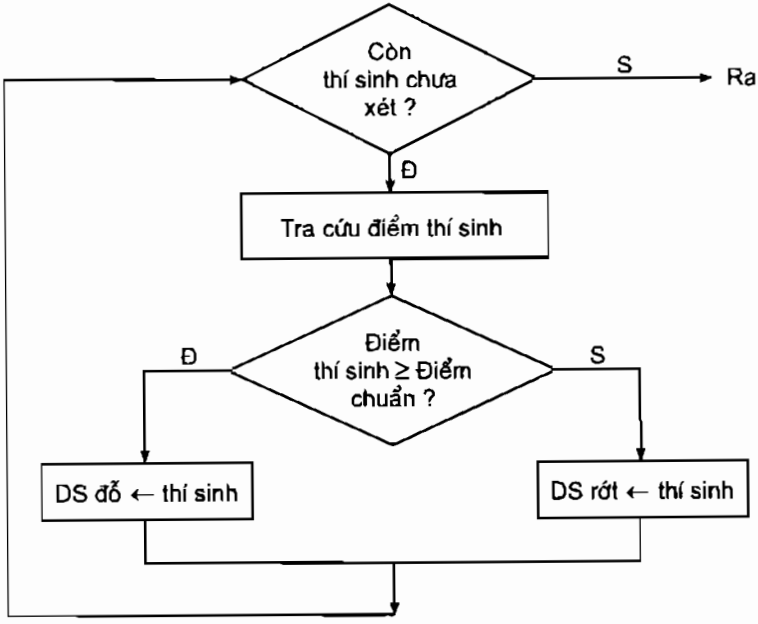
Nếu như BLD chỉ có một loại nút là các chức năng (tức là các hành động phải làm), thì sơ đồ khối lại có hai loại nút:

- nút hành động xử lý (hình chữ nhật) và
- nút kiểm tra điều kiện (hình thoi).

Nếu trong BLD một cung là một tuyến chuyển giao dữ liệu, thì trong sơ đồ khối một cung là một tuyến chuyển giao điều khiển (nghĩa là chuyển giao quyền thực hiện).

Như vậy nếu như các BLD chỉ tập trung diễn tả những việc phải làm là gì (với mối liên quan về dữ liệu giữa chúng), thì các sơ đồ khối lại có phần ôm đồm hơn, không những chỉ ra các việc phải làm, mà còn chỉ ra cách dẫn dắt thực hiện các việc đó. Chính vì sự ôm đồm đó mà nó không thích hợp để diễn tả các chức năng phức tạp và lớn.

Hình III.19 cho một thí dụ dùng một sơ đồ khối để đặc tả chức năng "Lập danh sách trúng tuyển".



Hình III. 19 Một sơ đồ khối

d) Các ngôn ngữ có cấu trúc

Ngôn ngữ có cấu trúc (cũng còn được gọi là mã giả) là một ngôn ngữ tự nhiên (chẳng hạn tiếng Việt) bị hạn chế:

- chỉ được phép dùng các câu đơn sai khiến hay khẳng định (thể hiện các lệnh hay các điều kiện).
- các câu đơn này được ghép nối nhờ một số từ khóa thể hiện các cấu trúc điều khiển chọn và lặp.

Như vậy ngôn ngữ có cấu trúc có những đặc điểm của một ngôn ngữ lập trình, song nó không chịu những hạn chế và quy định ngặt nghèo của các ngôn ngữ lập trình, cho nên được dùng thoải mái hơn. Tuy nhiên nó cũng không quá phóng túng như một ngôn ngữ tự do.

Dưới đây là đặc tả của chức năng "Lập danh sách trúng tuyển" ở dạng ngôn ngữ có cấu trúc.

Lặp Lấy một thí sinh từ kho các thí sinh
Tra cứu điểm của thí sinh
Nếu Điểm thí sinh ≥ Điểm chuẩn
Thì DS đỗ ← thí sinh
Không thì DSrớt ← thí sinh
Đến khi Hết thí sinh

§2. PHƯƠNG PHÁP PHÂN TÍCH CÓ CẤU TRÚC (SA)

Phương pháp phân tích SA (Structured Analysis) do De Macro và những người khác (như Yourdon, Constantine...) đưa ra từ những năm 70, nhưng vẫn còn phát huy tác dụng cho đến ngày nay, đặc biệt là với các hệ thống thông tin quản lý. Nó vẫn là nền tảng của những phần mềm trợ giúp phân tích và thiết kế nổi tiếng, như Designer 2000 của ORACLE chẳng hạn. Đặc điểm của phương pháp này là đơn giản, dễ theo, nhưng không quá sơ lược, mà cũng không cầu kỳ như một số phương pháp khác. Công cụ biểu diễn chính được sử dụng trong quá trình phân tích là biểu đồ luồng dữ liệu (BLD).

Mục đích của SA là tiến hành phân tích chức năng của hệ thống để thành lập một mô hình lôgic về chức năng của hệ thống mới, dưới dạng một BLD (hay đúng hơn là một tập hợp các BLD). Nó vận dụng ba kỹ thuật:

- Kỹ thuật phân mức (hay phân tích từ trên xuống);
- Kỹ thuật chuyển đổi BLD vật lý thành BLD lôgic;
- Kỹ thuật chuyển đổi BLD hệ thống cũ thành BLD hệ thống mới.

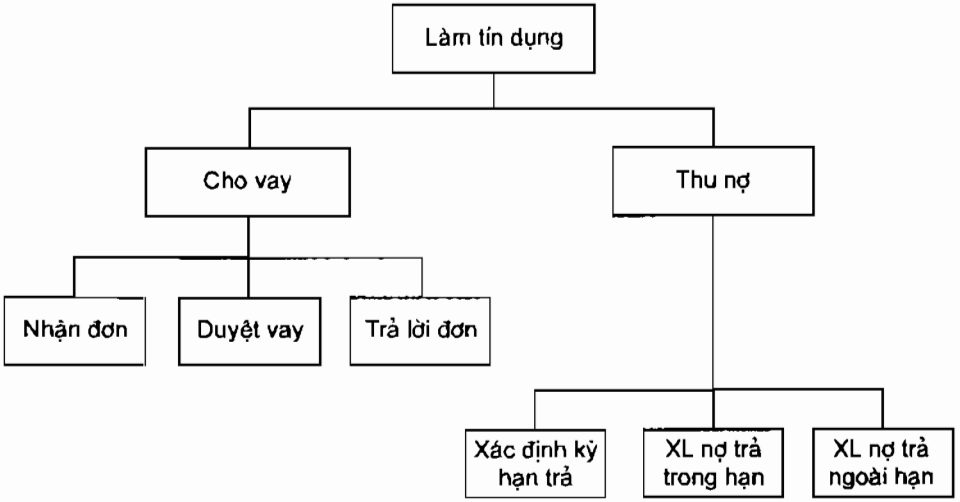
1. Kỹ thuật phân mức

Kỹ thuật phân mức hay còn gọi là "Phân tích từ trên xuống" (top-down analysis) tiến hành sự phân tích chức năng của hệ thống bằng cách đi dần từ một mô tả đại thể đến những mô tả chi tiết thông qua nhiều mức. Sự chuyển dịch từ một mức tới mức tiếp theo thực chất là sự phân rã mỗi chức năng ở mức trên thành một số các chức năng con ở mức dưới. Vậy đây là quá trình triển khai theo một cây, và chính vì vậy mà phương pháp này còn có tên là phương pháp phân tích có cấu trúc.

Có hai cách vận dụng kỹ thuật phân mức: Dùng biểu đồ phân cấp chức năng (BPC) và dùng biểu đồ luồng dữ liệu (BLD).

Với BPC (xem mục §1.2), thì phân tích trên xuống thực hiện bằng cách triển khai dần cây phân cấp từ gốc đến ngọn, lần lượt qua các tầng (mỗi tầng là một mức mô tả của hệ thống, bao gồm một tập hợp các chức năng). Để triển khai từ một tầng đến tầng tiếp theo, ta xem xét từng chức năng và đặt câu hỏi: để hoàn thành chức năng đó thì phải thực hiện các chức năng con nào. Nhờ đó ta phát hiện các chức năng thuộc tầng tiếp theo, mà mối liên quan với các chức năng ở tầng trên là quan hệ bao hàm (hay cha/con). Tên của mỗi chức năng (ở mỗi nút của cây) được chọn lựa để có thể phản ánh một cách ngắn gọn nội dung của chức năng đó. Chức năng ở gốc (mức 0) thể hiện nhiệm vụ tổng quát của hệ thống. Hình III. 20 cho một thí dụ diễn tả quá trình phân tích chức năng đối với hệ thống thông tin ở một cơ sở làm tín dụng.

Dễ thấy quá trình phân tích với BPC, mà thực chất là sự phân rã dần dần các chức năng, là tự nhiên và dễ làm. Tuy nhiên kết quả thu được có thể nói là nghèo nàn: Hệ thống chỉ được diễn tả bằng một tập hợp các chức năng rời rạc (ở một tầng của BPC). Vì vậy cách làm này chỉ thích hợp cho sự phân tích bước đầu, hoặc cho các hệ thống đơn giản.



Hình III. 20 BPC cho hệ thống tín dụng

Với BLD (xem mục §1.4) thì quá trình phân tích trên xuống lại là quá trình thành lập dần dần các BLD diễn tả các chức năng của hệ thống theo từng mức. Mỗi mức là một tập hợp các BLD:

- Mức 0, còn gọi là *mức bối cảnh* hay *khung cảnh*, chỉ gồm có một BLD, trong đó chỉ có một chức năng duy nhất (chức năng tổng quát của hệ thống) trao đổi các luồng thông tin với các đối tác.
- Mức 1, còn gọi là *mức đỉnh*, cũng chỉ gồm có một BLD, và các mức 2, 3, 4,..., mỗi mức gồm nhiều (> 1) BLD, được thành lập như sau:

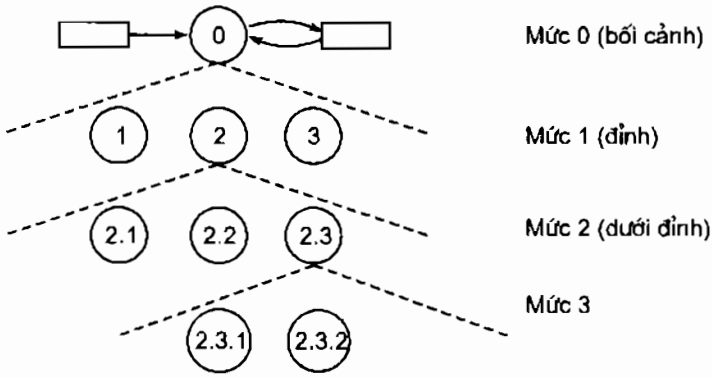
- Cứ mỗi chức năng ở mức trên, ta thành lập một BLD, ở mức dưới, gọi là BLD định nghĩa (hay giải thích), chức năng đó, theo cách sau:

  + Phân rã chức năng đó thành nhiều chức năng con;

  + Vẽ lại các luồng dữ liệu vào và ra chức năng trên, nhưng nay phải vào hay ra ở chức năng con thích hợp;

  + Nghiên cứu các quan hệ về dữ liệu giữa các chức năng con, nhờ đó bổ sung các luồng dữ liệu nội bộ hoặc các kho dữ liệu nội bộ.

- Các chức năng được đánh số theo ký pháp chấm, cho phép theo dõi vệt triển khai từ trên xuống



Hình III.21 Cách đánh số theo ký pháp chấm

Khi triển khai dần các mức như trên, ta phải tuân thủ một số quy tắc sau đây, gọi là các quy tắc về sự *tương hợp giữa các mức*:

- Khi định nghĩa một chức năng ở mức trên thành một BLD ở mức dưới thì các luồng dữ liệu vào và ra chức năng đó phải được bảo toàn, nghĩa là phải được vẽ lại đầy đủ trong BLD định nghĩa. Khi bảo toàn từ mức trên xuống mức dưới như vậy thì một luồng dữ liệu có thể phân rã thành nhiều luồng con, nếu cần.

- Các đối tác phải xuất hiện toàn bộ trong BLD bối cảnh, và không được phát sinh *mới* ở các mức dưới. Tuy nhiên có thể vẽ lại một đối tác ở mức dưới nếu thấy cần (chẳng hạn để thấy nơi xuất phát hoặc nơi đến của một luồng dữ liệu nào đó).

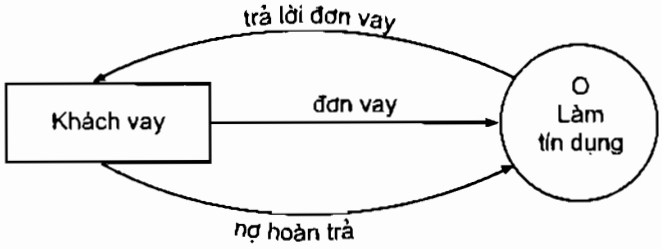
- Các kho dữ liệu không xuất hiện trong BLD bối cảnh. Tuy nhiên chúng sẽ dần dần phát sinh ở các mức dưới khi cần đến.

- Một đối tác hay một kho dữ liệu có thể được vẽ lặp lại, ở trong cùng một BLD hay ở BLD mức dưới, chỉ với mục đích là làm cho biểu đồ dễ đọc, dễ hiểu hơn (giúp thấy rõ nơi xuất phát hay nơi đến của các luồng dữ liệu, hoặc giúp cho các luồng dữ liệu đỡ chồng chéo), mà không được xem đấy là đối tác hay kho dữ liệu mới.

Trở lại thí dụ về hệ thống thông tin ở Cơ sở tín dụng thì quá trình phân tích trên xuống là như sau:

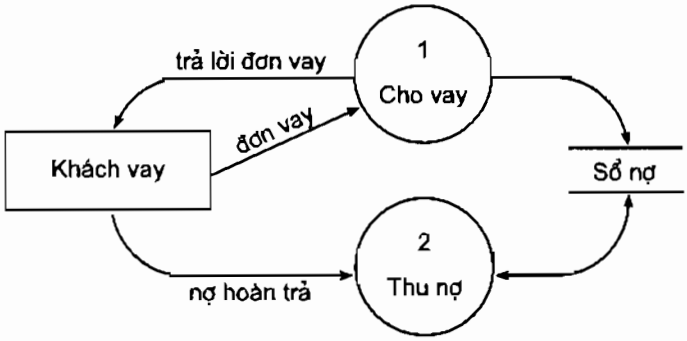
Mức O: Chức năng tổng quát của hệ thống là: Làm tín dụng. Đối tác của hệ thống là khách vay (cũng có thể còn có những đối tác khác như là Ngân hàng, chính quyền địa phương v.v..., song ta tạm thời lược bỏ cho đơn giản). Bổ

sung các luồng dữ liệu trao đổi giữa hệ thống và đối tác, ta có biểu đồ luồng dữ liệu bối cảnh như ở Hình III.22.



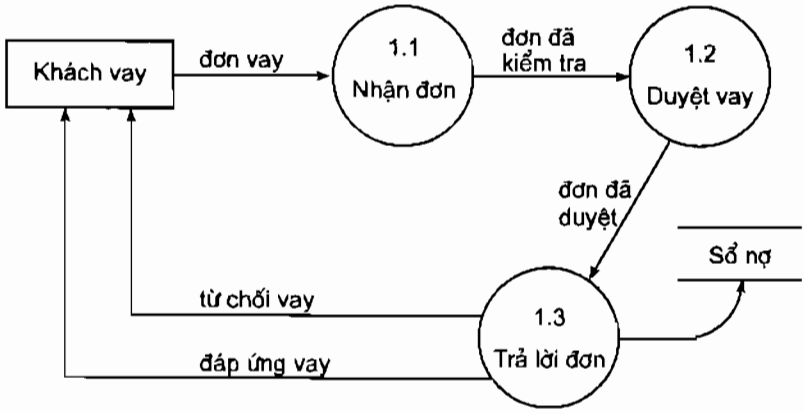
Hình III. 22 BLD bối cảnh

Mức 1: Chức năng 0 có thể phân rã thành hai chức năng con là Cho vay và Thu nợ. Ngoài ba luồng DL vào/ ra chức năng 0 được bảo toàn, thì ta thấy luồng thông tin trao đổi giữa hai chức năng Cho vay và Thu nợ là không trực tiếp, mà phải thông qua một kho dữ liệu, là cái Sổ nợ. Từ đó có BLD mức đỉnh như trong Hình III. 23.

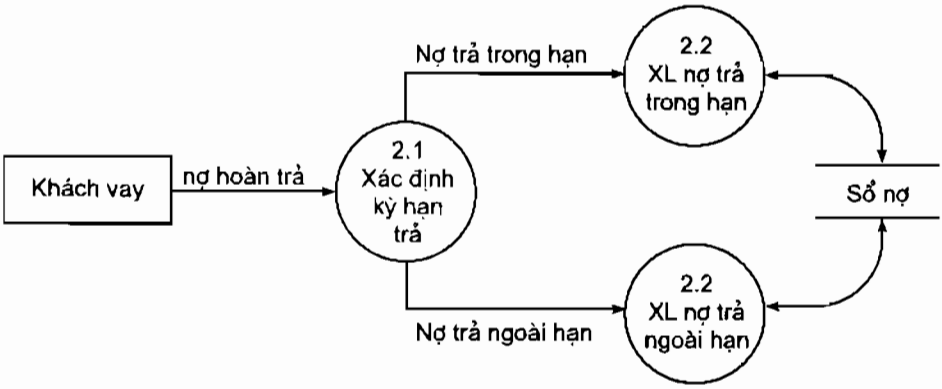


Hình III.23 BLD mức đỉnh

Mức 2: Chức năng 1 được phân rã thành ba chức năng: Nhận đơn, Duyệt vay và Trả lời đơn. Còn chức năng 2 thì được phân rã thành ba chức năng: Xác định kỳ hạn trả, xử lý nợ trả trong hạn và xử lý nợ trả ngoài hạn. Bảo toàn các luồng DL vào/ ra và thêm các luồng DL nội bộ, ta lập được hai BLD định nghĩa hai chức năng 1 và 2 như trong các Hình III. 24 và III.25.



Hình III.24 BLD định nghĩa chức năng 1 (Cho vay)



Hình III.25 BLD định nghĩa chức năng 2 (Thu nợ)

Chú thích:

+ Nếu cùng một hệ thống, mà ta tiến hành phân tích bằng cả hai cách: dùng BPC và dùng BLD thì đương nhiên giữa hai mô hình phải có mối liên quan ăn khớp: Luôn luôn phải có ánh xạ 1-1 giữa các chức năng trong BPC với các chức năng trong các BLD tương ứng. Có thể thấy điều đó khi ta so sánh Hình III.20 với các Hình III.22, III.23, III.24, III.25.

+ Quá trình triển khai trên xuống không thể kéo dài mãi, mà phải dừng sau một số mức. Ta quyết định dừng quá trình, khi có những biểu hiện sau:

- Các chức năng đã là khá đơn giản;
- Việc triển khai tiếp đã vượt ra ngoài câu hỏi "Làm gì?" và bắt đầu lạc sang câu hỏi "Làm như thế nào?";

- Số mức đã vào khoảng 7 ± 2 (tùy thuộc vào hệ thống là đơn giản hay phức tạp).

Khi dừng việc triển khai bằng BLD, thì với mỗi chức năng trong các BLD ở mức cuối cùng, ta phải cho một đặc tả trực tiếp (xem lại mục §1.5).

+ Quá trình triển khai đã làm phát sinh rất nhiều tên gọi: tên chức năng, tên luồng dữ liệu, tên kho dữ liệu, tên đối tác. Các tên đó rất dễ gây lầm lẫn cho các người dùng cũng như các người thiết kế sau này. Vậy phải lập một từ điển dữ liệu để giải thích các tên gọi đó. Nội dung và cấu trúc của từ điển dữ liệu sẽ được đề cập trong Chương IV.

2. Kỹ thuật chuyển đổi BLD vật lý thành BLD lôgic

Nhớ rằng mục đích của phân tích là đi tới một mô hình lôgic của hệ thống. Quá trình phân mức nói ở mục trên chỉ có thể dẫn ta vào những mô tả chi tiết, chứ không phải là đã đưa đến những mô tả lôgic. Bởi vì trong mô tả mà ta thu được (dưới dạng các BLD) thì vẫn còn lẫn các yếu tố vật lý vào đó. Nhất là khi sự mô tả mà ta thành lập lại dựa trên sự khảo sát một hệ thống đang tồn tại, đang hoạt động, thì các yếu tố vật lý, vốn đầy rẫy trong thực tế, sẽ có nhiều cơ hội lọt qua sự sàng lọc của người phân tích để đi vào các BLD được thành lập.

Có ba loại yếu tố vật lý có thể lẫn vào các BLD:

(1) Các yếu tố vật lý xuất hiện *tường minh trong ngôn từ hay hình vẽ* ở trong biểu đồ, như là:

- Các phương tiện, phương thức được dùng để thực hiện các chức năng (như máy tính, bàn phím, máy in, xử lý thủ công v.v...).
- Các giá mang thông tin (như đĩa từ, sổ sách, chứng từ trên giấy, đường điện thoại v.v...).
- Các tác nhân thực hiện chức năng (như giám đốc, kế toán viên, thủ kho v.v...).

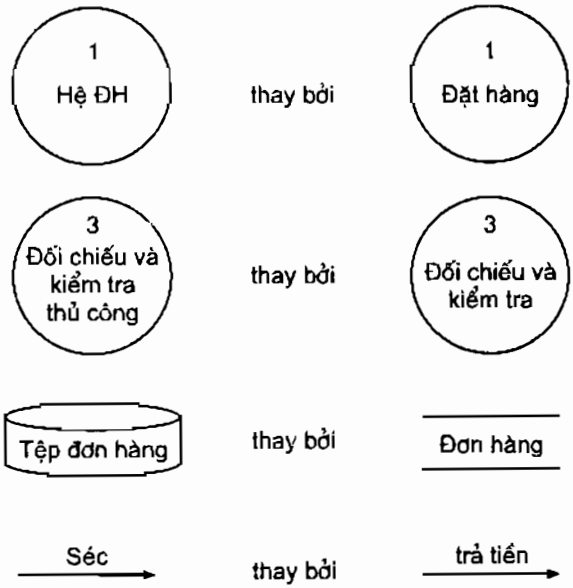
(2) *Các chức năng vật lý*, đó là những chức năng gắn liền với một công cụ hay một biện pháp xử lý nhất định, và sẽ không còn lý do tồn tại khi công cụ hay biện pháp đó bị thay đổi; chẳng hạn chức năng nhập dữ liệu vào máy tính sẽ không tồn tại nữa khi ta không dùng máy tính, và đó là một chức năng vật lý.

(3) *Cấu trúc vật lý*, ấy là cấu trúc chung của biểu đồ đang còn phản ánh trực tiếp cách bố trí, tổ chức hay cài đặt hiện tại, mà chưa phản ánh rõ bản chất

lôgic của hệ thống, chưa thoát ra ngoài mọi cách cài đặt cụ thể. Chẳng hạn ở hệ CƯVT ở nhà máy Z, có 3 tổ công tác là: Đặt hàng, Nhận/ Phát hàng và Đối chiếu. Do đó, một cách tự nhiên, BLD mức đỉnh được thành lập với 3 chức năng tương ứng với 3 tổ đó. Cấu trúc như vậy là cấu trúc chủ quan, vì nó bị áp đặt bởi một hình thức tổ chức công việc có thể là còn tạm thời và chưa hẳn đúng thực chất.

Các yếu tố vật lý nói trên xen lẫn vào BLD, làm cho BLD phản ánh không thật trúng bản chất của hệ thống, cần phải gạt bỏ chúng ra khỏi biểu đồ.

Để loại bỏ các yếu tố vật lý loại (1), ta chỉ cần loại bỏ ra khỏi biểu đồ các phần ngôn từ hay hình vẽ thể hiện phương tiện, giá mang thông tin hay tác nhân, và chỉ giữ lại sự diễn tả nội dung của chức năng hay của thông tin mà thôi. Chẳng hạn:



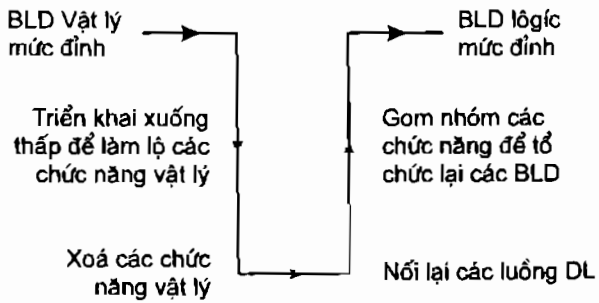
Để loại bỏ các yếu tố vật lý loại (2), tức là các chức năng vật lý, trước hết phải làm cho chúng xuất hiện trong biểu đồ đã. Thường thì các chức năng vật lý là các chức năng nhỏ, chưa xuất hiện ở các BLD mức trên. Vậy ta phải triển khai các BLD xuống các mức thấp. Khi phân rã mỗi chức năng thành chức năng nhỏ, ta sẽ tách được các chức năng vật lý rời khỏi các chức năng lôgic. Từ đó ta loại các chức năng vật lý ra khỏi biểu đồ.

Sau khi loại hết các chức năng vật lý, thì các chức năng còn lại là các chức năng lôgic và ta đang ở một mức thấp nào đó. Bấy giờ ta sẽ loại bỏ yếu tố vật lý

loại (3), tức là cấu trúc vật lý, bằng cách tổ chức lại các biểu đồ, từ dưới lên trên, xuất phát từ các chức năng lôgic nói trên, theo các bước như sau:

- Trước hết, do có các chức năng vật lý bị loại, một số luồng dữ liệu đã bị đứt quãng, ta phải tìm cách nối chúng lại cho liên tục.
- Tiếp đó, xem xét nội dung các chức năng (lôgic), tìm cách gom cụm các chức năng gần gũi và hợp tác với nhau trong một mục đích xử lý vào một chức năng lớn, cho dù trước đây chúng bị chia lìa bởi các lý do cài đặt. Làm như thế, đang ở mức dưới, ta lại thành lập được BLD ở mức trên. Và cứ thế tiếp tục, ta tổ chức lại các BLD ở các mức, cho đến mức đỉnh. Cấu trúc của các biểu đồ bấy giờ sẽ không còn mang tính chất vật lý nữa.

Tóm lại, từ BLD vật lý mức đỉnh, ta triển khai xuống các mức thấp, rũ bỏ các chức năng vật lý, rồi lại trở về các mức cao để chỉnh đốn lại cấu trúc của các biểu đồ. Rốt cục ta trở lại về mức đỉnh với một BLD lôgic. Hình III. 26 diễn tả lại một cách khái quát quá trình đó.

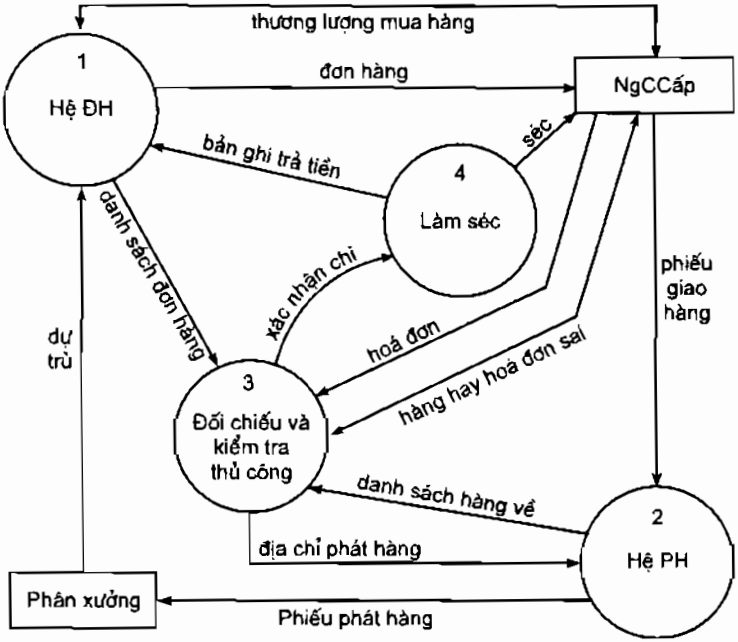


Hình III. 26 Quá trình rũ bỏ các yếu tố vật lý

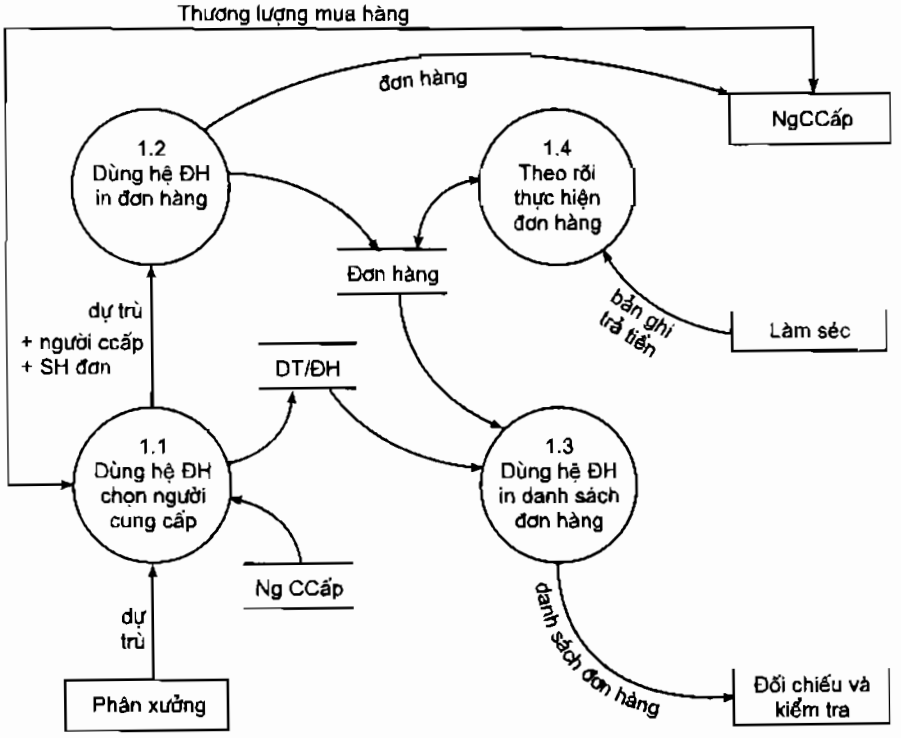
Thí dụ QL (tiếp theo) Trở lại hệ CƯVT ở nhà máy Z, và hãy lập một mô hình lôgic phản ánh hệ thống hiện tại.

Thoạt tiên, căn cứ vào các kết quả khảo sát, ta phản ánh hệ CƯVT hiện hành bằng một BLD vật lý mức đỉnh như trong Hình III. 27.

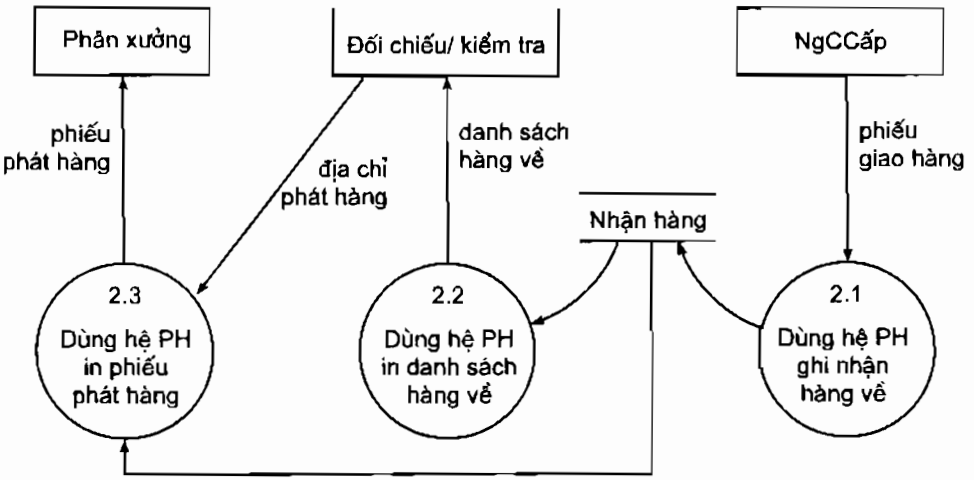
Sau đó triển khai thêm một mức (cũng vẫn căn cứ trên kết quả điều tra thực trạng), ta thu được ba BLD định nghĩa lần lượt các chức năng 1, 2, 3 như trong các Hình III. 28, III. 29 và III. 30. Riêng chức năng 4 (Làm séc) rất đơn giản nên không phải triển khai và chỉ đánh số lại là 4.1. Vả lại chức năng này thực ra là ở phòng Tài vụ, chứ không thuộc hệ CƯVT sau này ta sẽ loại nó ra khỏi phạm vi cài đặt.



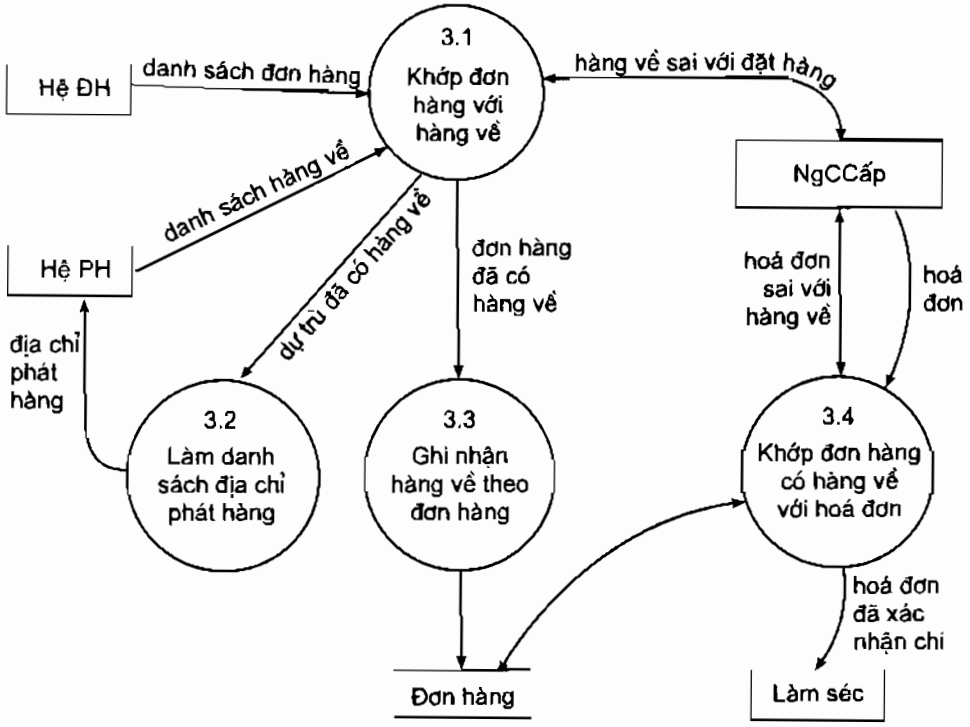
Hình III. 27 BLD vật lý mức đỉnh hệ CƯVT



Hình III. 28 BLD định nghĩa chức năng 1



Hình III. 29 BLD định nghĩa chức năng 2



Hình III. 30 BLD định nghĩa chức năng 3

Bây giờ ta loại bỏ các yếu tố vật lý. Trước hết là các yếu tố vật lý loại (1). Bỏ các cụm từ "Dùng hệ ĐH", "Dùng hệ PH" và "thủ công". Thay từ "in" bởi từ "làm", "Làm séc" bởi "Thanh toán" và "Séc" bởi "Trả tiền".

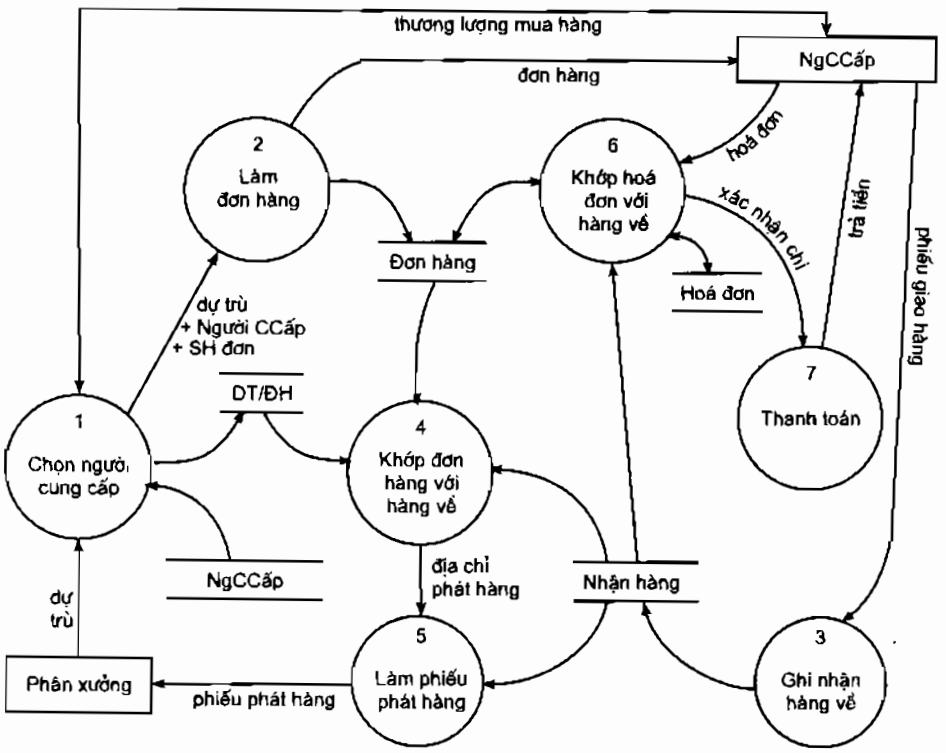
Tiếp đến là các yếu tố vật lý loại (2). Các chức năng 1.3 và 2.2 là chức năng vật lý, vì chúng chỉ làm nhiệm vụ đưa các thông tin từ các tệp DT/ĐH, Đơn hàng và Nhận hàng lên giấy. Chúng chỉ cần thiết trong phương thức tổ chức hiện tại mà thôi, do đó có thể loại đi.

Cuối cùng, xuất phát từ 9 chức năng còn lại ở mức 2 (1.1, 1.2, 1.4, 2.1, 2.3, 3.1 3.2, 3.3, 3.4), ta gom nhóm chúng theo sự gần gụi về mục đích xử lý (dù chúng vốn rải rác trên các biểu đồ khác nhau), và như vậy ta trở về lại mức đỉnh:

- Chức năng 1.1 trở thành 1 (Chọn người cung cấp);
- Chức năng 1.2 trở thành 2 (Làm đơn hàng);
- Chức năng 2.1 trở thành 3 (Ghi nhận hàng);
- Chức năng 3.1, 3.2, 3.3 gộp thành 4 (Khớp đơn hàng với hàng về);
- Chức năng 2.3 trở thành 5 (Làm phiếu phát hàng)
- Chức năng 3.4 và 1.4 gộp thành 6 (Khớp hóa đơn với hàng về);
- Chức năng 4.1 (vốn là 4), trở thành 7 (Thanh toán)

Kết quả là thu được BLD lôgic mức đỉnh như trên Hình III.31, trong đó một số luồng dữ liệu được vẽ lại, cho phép các chức năng lấy trực tiếp thông tin cần thiết từ các kho dữ liệu, không tính tới việc các kho dữ liệu đó đặt ở đâu trong thực tế.

Hãy so sánh BLD trên Hình III.31 với BLD trên Hình III.27, để thấy rằng nay thì chỉ có những chức năng gắn với mục đích xử lý mới được thể hiện, còn những chức năng gắn với công cụ và biện pháp đều đã bị bỏ đi. Mặt khác, phương thức tổ chức và các ranh giới chia cắt trong hiện trạng (thành 4 tổ) đã bị xóa nhòa. Chính nhờ vậy mà BLD lôgic trên Hình III.31 phản ánh rõ hơn bản chất của hệ thống hiện thời, trả lời một cách minh bạch câu hỏi: "Hệ thống hiện thời làm gì?".



Hình III.31 BLD lôgic mức đỉnh của hệ CƯVT (hiện trạng)

Chú thích: Các luồng khiếu nại về hàng sai hay hóa đơn sai tạm thời không vẽ trong Hình III.31 (và cả sau này) cho đỡ rườm rà.

3. Kỹ thuật chuyển từ BLD của HT cũ sang BLD của HT mới

Thường thì ít khi phải xây dựng mới hoàn toàn BLD lôgic của HT tương lai, mà chỉ xây dựng nó từ BLD của HT cũ, qua các bước thêm, bớt hay chỉnh đốn lại mà thôi.

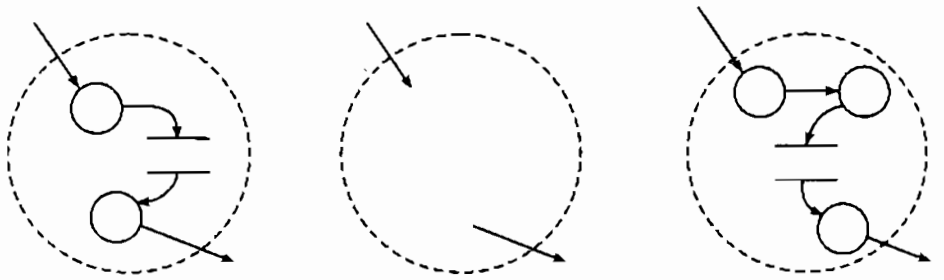
Để thực hiện sự chuyển đổi đó, trước hết cần xem lại các điểm phê phán đã đưa ra trước đây đối với HT cũ, cũng như xem lại các mục tiêu và ưu tiên đã đề xuất đối với HT mới. Nhờ sự phân tích chi tiết mà ta vừa tiến hành, nay có thể khẳng định lại, bổ sung hay sửa đổi các phê phán và các mục tiêu hay ưu tiên đó.

Căn cứ vào đó, đối chiếu với BLD lôgic của HT cũ, ta phát hiện trong đó có những chỗ thiếu sót, dư thừa hay cần sửa đổi lại. Khoanh từng vùng đó lại và gọi đó là các *vùng thay đổi*.

Đối với từng vùng thay đổi đó, ta tiến hành các bước biến đổi như sau:

- Xóa phần BLD bên trong vùng, song vẫn giữ lại các luồng vào/ra (các luồng đi qua ranh giới của vùng);
- Xác định chức năng tổng quát mới của vùng thay đổi;
- Thiết lập một trung tâm biến đổi mới (tức là một BLD) thực hiện chức năng tổng quát nói trên, gồm: các chức năng hợp thành, các kho dữ liệu cần thiết và các luồng dữ liệu liên kết các chức năng và các kho đó;
- Nối trung tâm biến đổi đó với các luồng vào/ra của vùng, nếu cần có thể bổ sung các chức năng biến đổi trung gian các luồng vào/ra đó, hoặc thêm luồng vào/ra mới.

Hình III.32 Tóm tắt các bước tiến hành trên.



(1) Khoanh một vùng thay đổi

(2) Xóa bên trong, giữ các luồng vào/ra

(3) Lập lại BLD bên trong khớp với các luồng vào/ra

Hình III.32 Các bước chỉnh sửa một vùng thay đổi

Thí dụ QL (tiếp theo): Ta hãy lập BLD lôgic cho HT mới. Như ta đã thấy ở mục §1.6 Chương II, thì 3 nhược điểm của hệ CƯVT cũ là:

(1) Thiếu một kho hàng dự trữ để giải quyết ngay các dự trù về các mặt hàng thông dụng;

(2) Chu trình quá lâu, do khâu chờ đợi tìm địa chỉ phát hàng;

(3) Kiểm tra không chặt, để xảy ra sai sót hàng - tiền luôn;

(4) Tốn nhân lực ở khâu đối chiếu và kiểm tra thủ công.

Các nhược điểm (2), (3), (4) đều bắt nguồn từ sự chia cắt các bộ phận, làm cách lế các kho thông tin, buộc phải tổ chức việc đối chiếu bằng tay, vừa chậm,

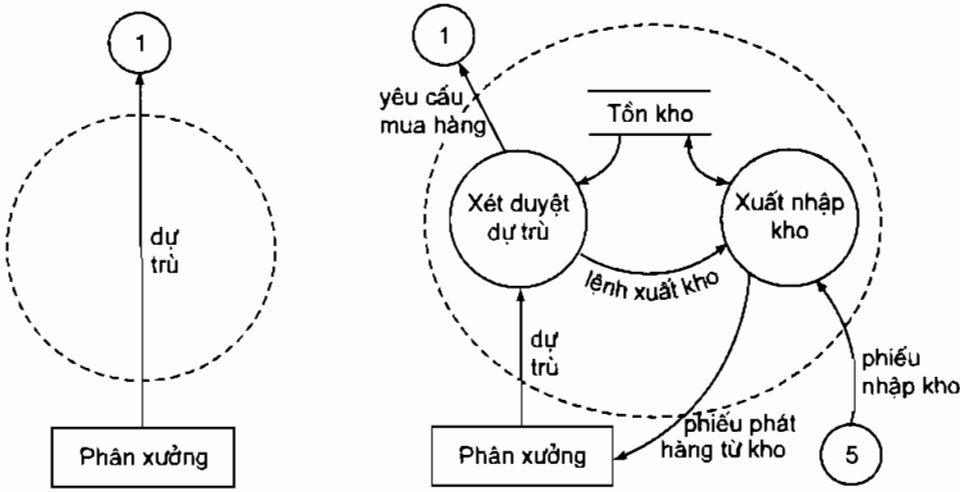
vừa tốn nhân lực. Nói cách khác các nhược điểm đó đều có nguồn gốc vật lý. Bởi vậy BLD lôgic của HT cũ ở Hình III.31 không còn thể hiện các nhược điểm đó, vì trong BLD lôgic thì các yếu tố vật lý đều đã bị xóa nhòa rồi.

Trái lại nhược điểm (1) là một thiếu sót lôgic. Vậy cần lập một vùng thay đổi trên BLD ở Hình III. 31 cắt ngang luồng "dự trù" từ Phân xưởng đến chức năng 1.

Chức năng tổng quát của vùng đó là quản lý kho hàng dự trữ để đáp ứng ngay các mặt hàng thông dụng cho các dự trù.

Vậy ở đây cần có một kho dữ liệu về các hàng tồn kho, một chức năng xuất nhập kho và một chức năng xét duyệt dự trù để xem dự trù có thể đáp ứng được từ hàng có trong kho hay phải chuyển sang đặt hàng. Chính chức năng này sẽ chủ động đưa ra yêu cầu đặt hàng cho kho mỗi khi có mặt hàng nào đó sụt xuống dưới ngưỡng quy định của nó.

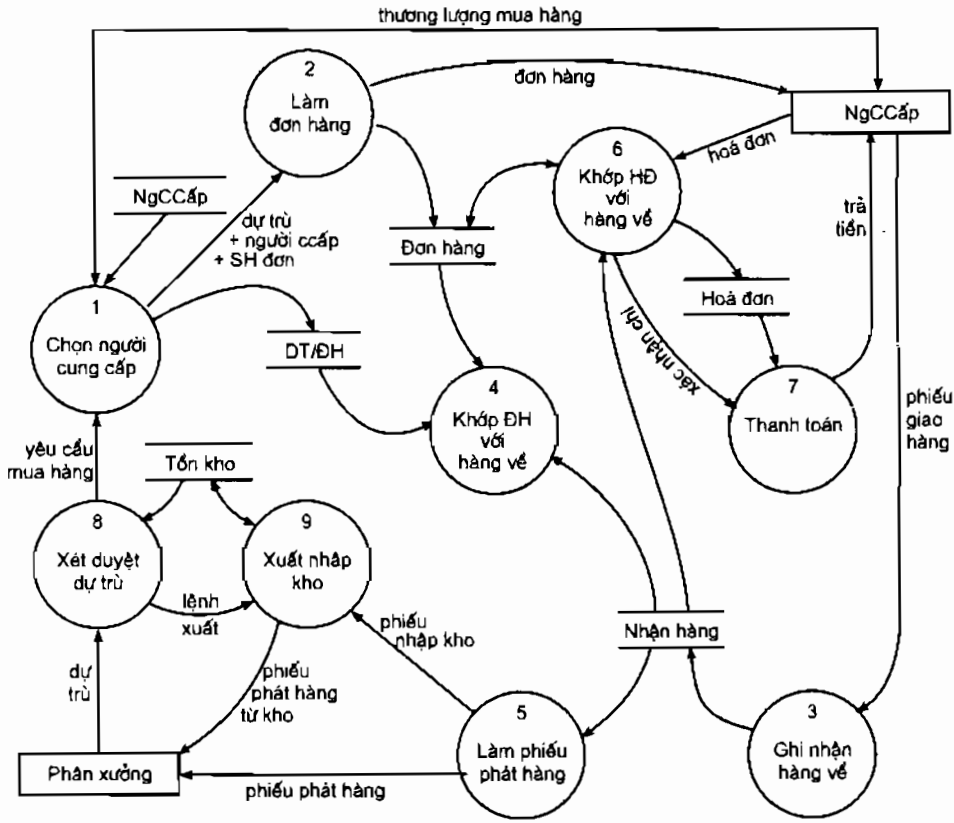
Hình III.33 cho vị trí của vùng thay đổi và nội dung bổ sung cho vùng thay đổi. Còn hình III.34 cho kết quả tổng hợp cuối cùng, tức là BLD lôgic (mức đỉnh) cho Hệ thống mới.



(a) Vị trí vùng thay đổi

(b) Nội dung bổ sung vào vùng thay đổi

Hình III.33 Thực hiện vùng thay đổi



Hình III.34 BLD lôgic của HT CƯVT mới

Chương IV
PHÂN TÍCH HỆ THỐNG VỀ DỮ LIỆU
(bước sơ bộ)

Mục đích của phân tích hệ thống về dữ liệu là lập *lược đồ khái niệm* về dữ liệu, làm căn cứ cho việc thiết kế cơ sở dữ liệu của hệ thống sau này. Sở dĩ nó còn được gọi là lược đồ khái niệm (để phân biệt với lược đồ vật lý), bởi vì khi cài đặt nó còn phải được chỉnh sửa ít nhiều cho phù hợp với các yêu cầu về thiết kế (ở giai đoạn này thì ta chưa thể đề cập đến các yêu cầu này).

Việc phân tích hệ thống về dữ liệu được tiến hành một cách độc lập với việc phân tích hệ thống về chức năng (đã đề cập ở Chương III) bởi vì ta muốn tập trung nghiên cứu cấu trúc tĩnh của dữ liệu (không phụ thuộc vào các xử lý, không phụ thuộc vào thời gian).

Nói vậy không phải là ta bỏ qua mối liên quan tất yếu giữa các xử lý và các dữ liệu, mà ta chỉ tạm hoãn việc xem xét mối liên quan này cho tới giai đoạn thiết kế, khi cần lập lược đồ vật lý của cơ sở dữ liệu.

Lược đồ khái niệm về dữ liệu được thành lập theo mô hình thực thể/liên kết (E/A), rồi được hoàn chỉnh theo mô hình quan hệ. Chương này chủ yếu trình bày mô hình E/A và trong chương tiếp sau sẽ nói về mô hình quan hệ. Song trước khi đề cập hai mô hình đó, ta hãy xét một vài phương tiện tuy thô sơ, nhưng cũng hay được dùng để diễn tả và quản lý các dữ liệu (đặc biệt là các dữ liệu phi số): đó là các mã và các từ điển dữ liệu.

§1 - MỘT SỐ PHƯƠNG TIỆN SƠ ĐẲNG ĐỂ DIỄN TẢ VÀ QUẢN LÝ DỮ LIỆU

1. Mã hóa các tên gọi

a) Vấn đề

Dữ liệu dùng trong hệ thống thường ở hai dạng: số và dãy ký tự (phi số). Về ý nghĩa thì dãy ký tự là tên của một đối tượng nào đó trong hệ thống.

Ta gọi *mã hóa* (codification) là việc gán một tên gọi vắn tắt (gọi là mã) cho một đối tượng nào đó. Các đối tượng trong hệ thống được đặt tên có thể là:

- các ứng dụng tin học khác nhau trong doanh nghiệp;
- các chức năng;
- các đơn vị xử lý;
- các chương trình;
- các tài liệu;
- các tệp dữ liệu;
- các thông tin trong các tài liệu và các tệp;
- các biến dùng trong các chương trình v.v...

b) Chất lượng cơ bản của mã hóa

Việc mã hóa phải cố gắng đạt một số yêu cầu về chất lượng như sau:

- Không nhập nhằng: Đó phải là một ánh xạ 1-1 từ tập các đối tượng *vào* tập các mã.
- Thích hợp với phương thức sử dụng:
  + Sử dụng cho người: mã phải dễ hiểu, dễ giải mã.
  + Sử dụng cho máy tính: mã phải được định nghĩa một cách chặt chẽ.
- Có khả năng mở rộng và xen thêm:
  + Mở rộng: bổ sung phía trên và phía dưới;
  + Xen thêm: bổ sung trong một thứ tự.

Để thực hiện khả năng xen thêm có thể dùng hai cách:

  + Nhẩy cóc theo một giá trị nhất định.
  + Nhẩy cóc theo một kết quả thống kê.

- Phải ngắn gọn, bởi vì mã càng dài thì việc xử lý càng khó khăn. Tuy nhiên chiều dài của mã lại ảnh hưởng tới khả năng mở rộng mã. Ví dụ với mã là 4 con số thì nhiều nhất có thể chỉ định 9999 đối tượng.
- Có tính gợi ý (diễn nghĩa): nhìn mã, con người có thể dễ đoán ra đối tượng. Chẳng hạn để mã hóa các thành phố thì:
  + Hà Nội được gán mã 29 (trong biển số xe) là kém gợi ý.

+ Trái lại, trong ngành hàng không, người ta chỉ cần dùng ba chữ cái để chỉ định các thành phố, mà vẫn giàu tính diễn nghĩa:

Hà Nội	có mã là	HAN
Băngcốc	-	BKK
Bombay	-	BOM
Nairobi	-	NAI

Chú ý: Rõ ràng là các yêu cầu nêu trên không thể thỏa mãn đồng thời được, vì chúng có thể loại trừ lẫn nhau.

c) Các kiểu mã hóa khác nhau

• *Mã hóa liên tiếp*: Dùng các số liên tiếp để trỏ các đối tượng.

Thí dụ: Mã hóa các khách hàng theo thứ tự thời gian:

001, 002, ..., 084,...

Ưu điểm:

- Không nhập nhằng (nếu không dùng lại các mã số đã bị loại);
- Đơn giản;
- Mở rộng phía sau được (nếu không hạn chế về độ dài).

Khuyết điểm:

- Không xen thêm được;
- Không gợi ý, vậy phải có một bảng tương ứng mã và đối tượng;
- Không phân nhóm.

Lưu ý: Không nên dùng lại một mã đã dùng, dù nó đã bị loại.

• *Mã hóa theo lát*: Dùng từng lát cho từng loại đối tượng. Trong mỗi lát, thường dùng kiểu mã hóa liên tiếp.

Thí dụ: Các đối tượng là các hàng ngũ kim.

+ 0001 - 0999 : hàng ngũ kim bé, trong đó

0001 - 0099 : các loại vít

0100 - 0299 : các loại ê-cu

0300 - 0499 : các loại bu-lông

0500 - 0599 : các loại đinh

⋮

+ 1000 - 1999 : các chi tiết bằng kim loại, trong đó

1000-1099 : các sắt chữ U

⋮

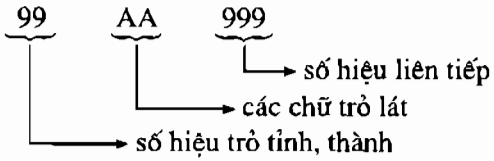
Ưu điểm:

- Không nhập nhằng (nếu các lát là tách rời, tức là không có đối tượng thuộc vào hai lát khác nhau);
- Đơn giản;
- Mở rộng và xen thêm được.

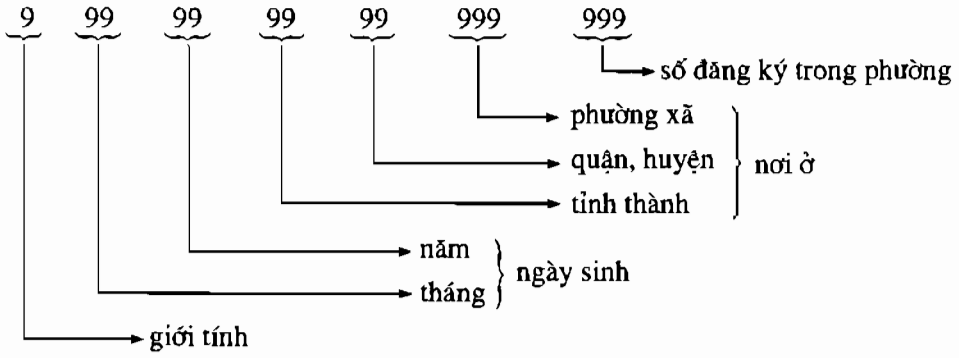
Khuyết điểm: Vẫn cần dùng bảng tương ứng.

- *Mã hóa phân đoạn*: Mã được phân thành nhiều đoạn, mỗi đoạn mang một ý nghĩa riêng:

Thí dụ: - số đăng ký xe máy:



- số bảo hiểm xã hội cho từng cá nhân:



Ưu điểm: - Không nhập nhằng;

- Mở rộng và xen thêm được (nếu mỗi đoạn còn chỗ);

- Dùng phổ biến;

- Cho phép thiết lập các kiểm tra gián tiếp (Ví dụ: số bảo hiểm có khớp với các thông tin tương ứng ghi trong chứng minh thư không?).

Khuyết điểm:

- thường quá dài;

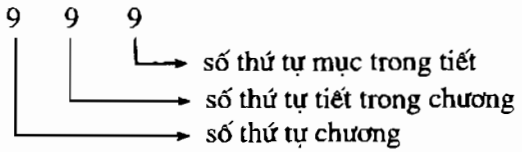
- thao tác nặng nề khi mã có quá nhiều đoạn;

- vẫn có thể bị bão hòa;

- mã không cố định (ví dụ đổi nơi ở thì phải đổi số bảo hiểm).

- *Mã hóa phân cấp*: Cũng là phân đoạn, song mỗi đoạn trỏ một tập hợp các đối tượng và các tập hợp đó bao nhau theo thứ tự từ trái qua phải.

Ví dụ: Đánh số các mục trong một cuốn sách



Ưu khuyết điểm: như mã hóa phân đoạn.

Thêm ưu điểm: tìm kiếm một đối tượng dễ dàng, bằng cách lần theo đoạn từ trái qua phải (tương ứng với một sự tìm kiếm trên cây).

- *Mã hóa diễn nghĩa*: Gán một tên ngắn gọn, nhưng hiểu được cho từng đối tượng.

Ví dụ: # HOADON là số hiệu hóa đơn.

Ưu điểm: Tiện dùng cho xử lý thủ công.

Khuyết điểm: Không giải mã được bằng máy tính.

2. Từ điển dữ liệu

a) Mục đích

Từ điển dữ liệu là một tư liệu tập trung về mọi tên gọi của mọi đối tượng được dùng trong hệ thống trong cả các giai đoạn Phân tích, Thiết kế, Cài đặt và Bảo trì.

Chẳng hạn:

- Ở mức lôgic, có:
  - các luồng dữ liệu, các giao dịch, các sự kiện,
  - các chức năng xử lý,
  - các thực thể,
  - các thuộc tính, v.v...
- Ở mức vật lý, có:
  - các tệp
  - các chương trình
  - các chương trình con, môđun, thủ tục v.v...

Từ điển dữ liệu là cần thiết đặc biệt cho quá trình triển khai các hệ thống lớn, có đông người tham gia. Nó cho phép:

* Trong phân tích và thiết kế: quản lý tập trung và chính xác mọi thuật ngữ và các mã dùng trong hệ thống, kiểm soát được sự trùng lắp, đồng nghĩa hay đồng âm dị nghĩa v.v...

* Trong cài đặt: người cài đặt hiểu được chính xác các thuật ngữ từ kết quả phân tích và thiết kế.

* Trong bảo trì: Khi cần thay đổi, thì phát hiện được các mối liên quan, các ảnh hưởng có thể nảy sinh. Ví dụ đổi một tên, thì biết rõ rằng tên đó được dùng ở những nơi nào để thay đổi.

b) Các hình thức thực hiện từ điển dữ liệu

Từ điển dữ liệu có thể được thực hiện và duy trì theo hai cách:

- Bằng tay: Đó là một tập tài liệu (như một từ điển thông thường) thành lập bởi người thiết kế và sau đó được duy trì và cập nhật bởi người quản trị hệ thống.
- Bằng máy tính: Dùng một hệ mềm, cho phép dễ dàng thành lập, thay đổi. Trong trường hợp này cần có một ngôn ngữ đặc tả thích hợp, thuận tiện cả cho người và cho máy tính trong việc miêu tả cấu trúc của các dữ liệu phức hợp.

Cũng như từ điển thông thường, từ điển dữ liệu là một tập hợp các *mục từ*, mỗi mục từ tương ứng với một tên gọi kèm với các giải thích đối với nó. Thường thì mỗi mục từ được chép trên một tờ giấy rời cho dễ sắp xếp.

c) Nội dung các mục từ

Trong mục từ, ngoài tên gọi và các tên đồng nghĩa thì phần giải thích thường đề cập đến bốn loại đặc điểm:

- Đặc điểm về cấu trúc: là nguyên thủy (đơn) hay phức hợp (nhóm).
- Đặc điểm về bản chất: là liên tục hay rời rạc.
- Đặc điểm về chi tiết: miền giá trị, đơn vị đo, độ chính xác, độ phân giải, số lượng, tần số, mức ưu tiên...
- Đặc điểm về liên hệ: từ đâu và đến đâu, đầu vào và đầu ra, dùng ở đâu...

Ngoài ra còn có thể có ghi chú hay lời bình.

Tuy nhiên, nội dung của các mục từ thường thay đổi theo loại của đối tượng mang tên gọi. Ta thường phân biệt các loại sau:

- Luồng dữ liệu;
- Kho dữ liệu (hay tệp dữ liệu);
- Dữ liệu sơ cấp (hay phần tử dữ liệu);
- Chức năng xử lý (hoặc chương trình, môđun).

Các hình từ IV.1 đến IV.4 cho một số thí dụ về cách trình bày các mục từ với các loại khác nhau.

		Định nghĩa luồng dữ liệu
Tên luồng dữ liệu	:	Hóa đơn
Tên đồng nghĩa	:	Hóa đơn kiêm phiếu thu
Vị trí (Từ/Đến)		
Từ	:	Lập hóa đơn
Đến	:	Giải quyết bán hàng theo tuần
Hợp thành	:	Tên khách hàng
		Ngày hóa đơn
		Ngày
		Tháng
		Năm
		Các khoản bán hàng
		Tên mặt hàng
		Số lượng
		Thành tiền
Giải thích	:	Giải trình tiền trả cho một đơn mua hàng
Lập ngày 10/10/96		Bởi : N.V.B

Hình IV.1 Mục từ cho một luồng dữ liệu

Định nghĩa dữ liệu sơ cấp		
Tên dữ liệu sơ cấp	:	Ngày mở tài khoản
Mô tả	:	là ngày mà một tài khoản của khách hàng bắt đầu hoạt động
Từ đồng nghĩa	:	Ngày TK
Hợp thành	:	Ngày + Tháng + Năm
Bản ghi, tệp liên quan	:	tệp khách hàng
Các xử lý có liên quan:		Biên tập đơn hàng
		Xây dựng tệp khách hàng
Đặc điểm dữ liệu	:	số ký tự 6, kiểu N
Các giá trị (miền có thể)	:	khuôn dạng DDMMYY năm không trước 90, ngày phải trước ngày hiện tại
Lập ngày 10/10/96		Bởi : N.V.B

Hình IV.2 Mục từ cho một dữ liệu sơ cấp

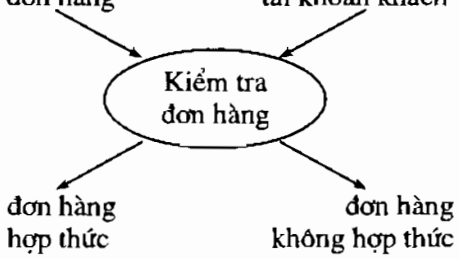
Định nghĩa tệp		
Tên tệp	:	Nhân viên
Mô tả	:	Chứa mọi thông tin về các nhân viên trong cơ quan
Từ đồng nghĩa	:	Không
Hợp thành	:	Mã số NV
		Tên NV
		Ngày bắt đầu công tác
		Lương
		Phòng
		{ Số hiệu đồ án + Người phụ trách đồ án } Từ 0 - 3
Tổ chức	:	Tuần tự theo mã số NV
Các xử lý liên quan:		Cập nhật nhân viên
		Tìm kiếm nhân viên
Lập ngày 10/10/96		Bởi : N.V.B

Hình IV.3 Mục từ cho một tệp dữ liệu

Định nghĩa chức năng xử lý

Tên chức năng : Kiểm tra đơn hàng

Lưu đồ :



Mô tả : Kiểm tra và biên tập một đơn hàng từ khách hàng tới, đối chiếu với tài khoản của khách, đưa ra đơn hàng hợp thức để xử lý tiếp, hoặc đơn hàng không hợp thức để trả lại cho khách.

Từ đồng nghĩa : Không

Vào : đơn hàng, tài khoản của khách

Ra : đơn hàng hợp thức/đơn hàng không hợp thức

Lập ngày 10/10/96 Bởi : N.V.B

Hình IV.4 Mục từ cho một chức năng xử lý

Trong mỗi mục từ, phần định nghĩa dữ liệu (hợp thành, giá trị...) thường được viết một cách ngắn gọn dựa vào một số quy ước như sau:

- Đối với các dữ liệu phức hợp, sự hợp thành của nó được thực hiện bởi các cấu trúc tuần tự, chọn, lặp. Có nhiều cách thể hiện các cấu trúc đó:

* Dùng các từ khóa (tiếng Anh hay tiếng Việt):

- tuần tự: dữ liệu A IS dữ liệu P
  AND dữ liệu Q
  AND dữ liệu R
- lặp: dữ liệu B IS ITERATION OF dữ liệu S
- chọn: dữ liệu C IS EITHER dữ liệu T

OR dữ liệu U

OR dữ liệu V

Thí dụ: Địa chỉ IS Tên

AND Số nhà

AND Đường, phố

AND Quận, huyện

AND Tỉnh, thành

Tệp nhân viên IS ITERATION OF

Bản ghi nhân viên

Giao dịch khách hàng IS EITHER Khách đóng tiền

OR Khách lấy hàng

* Dùng ký pháp EBNF (Extended Backus Naur Form):

: :	có nghĩa là	:	Hợp thành bởi
+	-	:	tuần tự
{...}	-	:	lặp
(...|...)	-	:	chọn một trong (hoặc có loại trừ)
<...|...>	-	:	ít nhất một trong (hoặc không loại trừ)
[...]	-	:	tùy chọn (không hay một lần)
...	-	:	chú thích

Thí dụ: Đơn hàng : : * Các thông tin về đặt hàng *

Tên khách hàng +

Địa chỉ khách hàng +

Số hiệu đơn hàng +

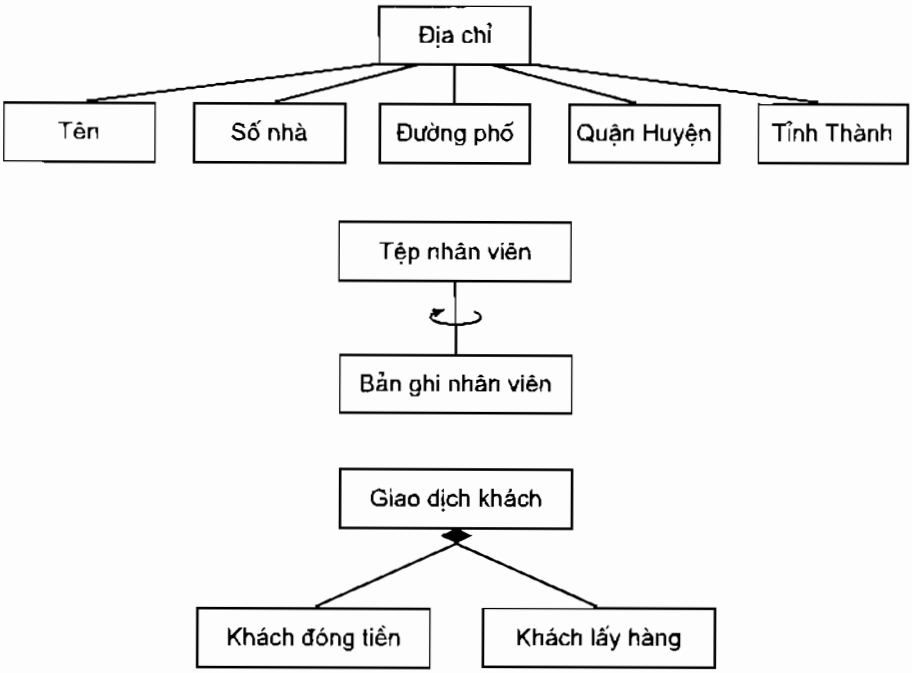
{ Tên mặt hàng +
Mã hàng +
Đơn giá +
Số lượng } +

Tổng tiền

Giao dịch khách hàng : : (Khách đóng tiền | khách lấy hàng)

* Dùng đồ thị:

Thí dụ:



- Đối với các dữ liệu đơn (sơ cấp): Định nghĩa theo các giá trị mà nó có thể nhận (miền giá trị)

Thí dụ:

Giá trị	*ý nghĩa*
AA	lớn
A	tốt
B	chấp nhận được
C	kém
D	không chấp nhận được
DD	dứt khoát loại trừ

Ghi chú: Phương pháp SA (của De Marco) lấy biểu đồ luồng dữ liệu làm mô hình về chức năng, và lấy từ điển dữ liệu làm mô hình về dữ liệu. Mặc dù từ điển dữ liệu là có ích cho quá trình triển khai hệ thống, song lấy nó làm mô

hình dữ liệu, để làm căn cứ cho việc thiết kế cơ sở dữ liệu sau này, thì ta dễ nhận thấy là quãng đường để lại còn quá xa, còn quá mịt mù. Nhược điểm chính của từ điển dữ liệu là nó thiếu tính hình thức (không chặt chẽ, kém trừu tượng) và nhất là nó không phản ánh được các mối liên quan vốn có giữa các dữ liệu. Bởi vậy, thay vì từ điển dữ liệu, ta sẽ lấy mô hình thực thể/liên kết (E/A) hay mô hình quan hệ (trình bày ở phần tiếp sau) làm mô hình dữ liệu của giai đoạn Phân tích.

§2. MÔ HÌNH THỰC THỂ/ LIÊN KẾT

Mô hình Thực thể/ Liên kết (Entity/ Association Model) là mô hình dữ liệu do P.P. Chen đưa ra năm 1976 và sau đó được dùng khá phổ biến trên thế giới.

Nó có ưu điểm là khá đơn giản và gần với tư duy trực quan. Khi xem xét các thông tin, người ta thường gom cụm chúng xung quanh các vật thể. Chẳng hạn các thông tin về tên, tuổi, địa chỉ, chiều cao, cân nặng... được gom cụm với nhau xung quanh một người, trong khi các thông tin về số đăng ký, nhãn mác, kiểu dáng, màu sơn, dung tích xy lanh... lại được gom với nhau xung quanh một xe máy. Mô hình thực thể/ liên kết mô tả tập hợp các dữ liệu dùng trong một hệ thống theo cách gom cụm như vậy.

Ở đây, ta sẽ dùng mô hình thực thể/liên kết trong bước đầu mô hình hóa về dữ liệu. Bước tiếp sau, ta sẽ cải thiện mô hình đó nhờ áp dụng mô hình quan hệ.

1. Mô hình thực thể/ liên kết kinh điển

Trước tiên ta sẽ giới thiệu trong mục này mô hình thực thể / liên kết (viết tắt là E/A) kinh điển. Đó là dạng ban đầu của mô hình, và cũng đã từng được ứng dụng nhiều (chẳng hạn dùng trong phương pháp MERISE). Ở các mục tiếp sau ta sẽ giới thiệu một mô hình E/A mở rộng, thuận theo xu hướng mô hình hóa hướng đối tượng của những năm 90; rồi lại giới thiệu một mô hình E/A hạn chế (dùng trong ORACLE) gần với mô hình quan hệ ở dạng chuẩn 3 và do đó cũng rất thuận tiện cho bước chuyển sang thiết kế.

a) Các khái niệm của mô hình E/A

Mô hình E/A xuất phát từ ba khái niệm cơ bản: thực thể, liên kết và thuộc tính.

• *Các thực thể:*

Một thực thể (entity) là một vật thể cụ thể hay trừu tượng, tồn tại thực sự và khá ổn định trong thế giới thực, mà ta muốn phản ánh nó trong hệ thống thông tin.

Thí dụ: Thực thể cụ thể như:

Khách hàng Nguyễn Văn Ân

Đơn hàng số 37458

Thực thể trừu tượng như:

Khoa Công nghệ thông tin

Tài khoản số 49578

• *Các thuộc tính*

Thuộc tính (property hay attribute) là một giá trị dùng để mô tả một khía cạnh nào đó của một thực thể.

Thí dụ: Tuổi của Nguyễn Văn Ân là 45

Tổng tiền của đơn hàng 37458 là 250.000đ.

Giá trị thuộc tính thường được cho kèm theo một tên (tuổi : 45, tổng tiền: 250.000đ). Tên đó thực chất là tên chung của mọi giá trị có thể chọn lựa để mô tả một khía cạnh nhất định của các thực thể (tuổi: 45, tuổi: 20, tuổi: 14...). Gọi tên đó là một *kiểu thuộc tính* (property type).

Thí dụ: Tuổi, tổng tiền... là các kiểu thuộc tính.

Ta gọi *kiểu thực thể* (entity type) là một tập hợp các thực thể được mô tả bởi cùng một tập hợp các kiểu thuộc tính và biểu diễn cho một lớp tự nhiên các vật thể trong thế giới thực.

Thí dụ: Kiểu thực thể khách hàng được mô tả bằng các kiểu thuộc tính: tên, địa chỉ, số tài khoản.

Một hay một tập kiểu thuộc tính của một kiểu thực thể được gọi là một *khóa* nếu giá trị của nó cho phép ta phân biệt các thực thể với nhau. Thí dụ: Số tài khoản và Mã hàng lần lượt là khóa của các kiểu thực thể Tài khoản và Mặt hàng. Còn với kiểu thực thể Học sinh, ta có thể lấy Tên và Ngày sinh làm khóa. Nếu khóa chỉ gồm một kiểu thuộc tính duy nhất thì ta gọi thuộc tính đó là một *định danh* (identifier). Tên định danh thường viết với các tiền tố ID, #, SH hay Mã.

Thí dụ: ID Nhân viên
Nhân viên
SH Nhân viên
Mã Nhân viên

Có hai ràng buộc phải được thỏa mãn đối với các kiểu thuộc tính:

(i) Giá trị duy nhất : mỗi thuộc tính của một thực thể có thể lấy một và chỉ một giá trị duy nhất (không phải là một dãy hay một tập hợp các giá trị).

(ii) Giá trị sơ đẳng: Giá trị thuộc tính không thể chia tách thành các thành phần nhỏ hơn (tức là không thể định nghĩa một kiểu thuộc tính từ các kiểu thuộc tính khác, bằng cách hợp thành hay thu gọn).

Thí dụ:

- Kiểu thuộc tính Tên các con của kiểu thực thể Nhân viên là không chấp nhận được vì với mỗi nhân viên, giá trị thuộc tính này có thể là không duy nhất (một dãy tên các con).

- Không thể định nghĩa kiểu thuộc tính Địa chỉ khách hàng như là hợp thành các kiểu thuộc tính Số nhà, Đường phố, Tỉnh thành (mà phải xem nó là một xâu ký tự duy nhất).

Các ràng buộc nói trên được đưa vào là nhằm làm cho các miền giá trị của các kiểu thuộc tính trở nên rất đơn giản, và như thế dễ chuyển sang cài đặt với các hệ QT CSDL quan hệ: Tuy nhiên các ràng buộc đó (của mô hình E/A kinh điển) sẽ làm giảm sự thoải mái, tự nhiên trong mô hình hóa thực tại.

• *Các liên kết*

Một liên kết (association) là một sự gom nhóm các thực thể trong đó mỗi thực thể có một vai trò nhất định.

Thí dụ:

- Khách hàng Ân đã giao nộp đơn hàng 3428.

- Đơn hàng 3428 đặt mua các mặt hàng 34 và 78

- Anh Liên là học trò của thầy Ất.

Một *kiểu liên kết* (association type) là một tập hợp các liên kết có cùng ý nghĩa. Một kiểu liên kết là được định nghĩa giữa nhiều kiểu thực thể. Số các kiểu thực thể tham gia vào kiểu liên kết gọi là số ngôi của kiểu liên kết. Tên của kiểu liên kết thường được chọn là một động từ (chủ động hay bị động) phản ánh ý nghĩa của nó.

Thí dụ:

- Kiểu liên kết Giao nộp giữa kiểu thực thể Khách hàng và kiểu thực thể Đơn hàng (2 ngôi).

- Kiểu liên kết Đặt mua giữa kiểu thực thể Đơn hàng và kiểu thực thể Mặt hàng (2 ngôi).

- Kiểu liên kết Dạy giữa kiểu thực thể Thầy và kiểu thực thể Trò (2 ngôi).

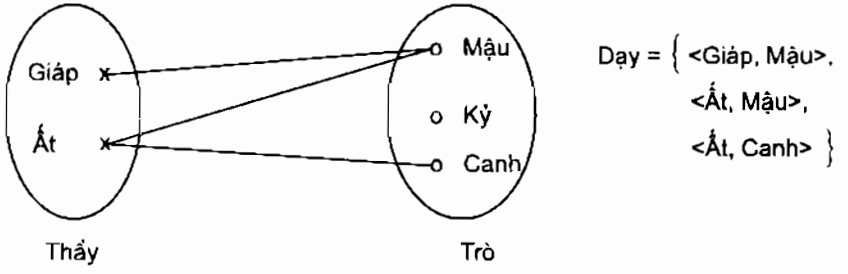
Chú thích:

- Tên liên kết thường chỉ phản ánh ý nghĩa liên kết theo một chiều. Chẳng hạn: Thầy dạy Trò, chứ không phải Trò dạy Thầy.

- Một kiểu liên kết n ngôi R giữa các kiểu thực thể T_1, T_2, ..., T_n thực chất là một quan hệ $R \subseteq T_1 \times T_2 \times ... \times T_n$ (theo định nghĩa toán học), tức là một tập hợp các bộ-n có dạng $< t_1, t_2, ..., t_n >$, với $t_i \in T_i$, i = 1,n. Có thể minh họa điều này (dưới dạng đồ họa) như trong Hình IV.5.

Trong một kiểu liên kết hai ngôi $R \subseteq T_1 \times T_2$, ta gọi *ứng số* (multiplicity) của R về phía T_1 (hoặc T_2) là một cặp số m..n trong đó m và n là các số tối thiểu và tối đa các thực thể trong T_1 (tương ứng T_2) có thể liên kết (qua R) với *một* thực thể trong T_2 (tương ứng T_1).

Thí dụ: Với kiểu liên kết Dạy cho trong Hình IV.5, thì ứng số về phía Thầy là: 0..2 (số thầy mà một trò có thể học), và phía Trò là 1..2 (số trò mà một thầy có thể dạy).



Hình IV. 5 Kiểu liên kết Dạy giữa hai kiểu thực thể Thầy và Trò

Các giá trị ứng số thường dùng là:

1	một và chỉ một (tức là 1..1)
0..1	không hay một

m..n	từ m tới n (m, n là các số tự nhiên)
0.. * hay *	từ không tới nhiều
1.. *	từ một tới nhiều.

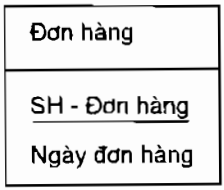
Cuối cùng, cần nhận xét thêm rằng đôi khi liên kết cũng có thể có thuộc tính. Ví dụ: Thầy Giáp dạy trò Mậu môn Toán vào năm 1999. Vậy kiểu liên kết Dạy có thể khắc họa thêm bởi các kiểu thuộc tính Môn dạy và Năm dạy.

Tóm lại, mô hình E/A kinh điển cho phép mô hình hóa một thế giới thực dưới dạng một tập hợp các kiểu thực thể, mỗi kiểu này được định nghĩa bởi một tập hợp các kiểu thuộc tính. Các kiểu thực thể được kết nối với nhau bởi các kiểu liên kết; các kiểu liên kết này lại có thể được định nghĩa bởi một tập hợp các kiểu thuộc tính và một bộ các ứng số.

b) Biểu diễn đồ họa các khái niệm của mô hình E/A

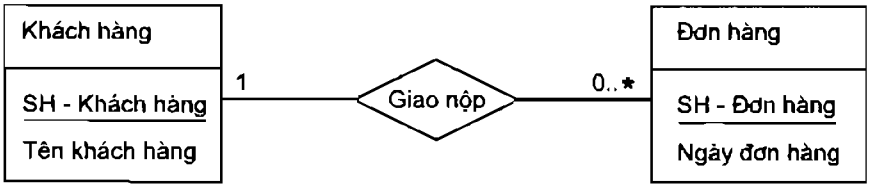
Để dễ nhận thức và trao đổi, mô hình E/A thường được biểu diễn dưới dạng một đồ thị, trong đó các nút là các kiểu thực thể, còn các cung là các kiểu liên kết. Đồ thị đó được gọi là *biểu đồ E/A* (E/A diagram) và được lập như sau:

Một kiểu thực thể được biểu diễn bởi một hình chữ nhật (Hình IV.6), gồm 2 ngăn: ngăn trên chứa tên của kiểu thực thể, ngăn dưới chứa danh sách các kiểu thuộc tính của nó. Tên kiểu thực thể thường là một danh từ (trỏ vật thể). Các kiểu thuộc tính hợp thành khóa của kiểu thực thể được gạch dưới, và đặt lên đầu danh sách (tuy nhiên khóa không nhất thiết phải làm rõ khi quá trình phân tích đang dở dang).



Hình IV. 6 Biểu diễn đồ họa của một kiểu thực thể

Một kiểu liên kết được biểu diễn bởi một hình thoi (Hình IV.7), được nối bằng nét liền tới các kiểu thực thể tham gia liên kết. Trong hình thoi viết tên kiểu liên kết (tên này có thể khuyết, nếu không cần làm rõ). Như trên đã nói tên kiểu liên kết thường là một động từ (chủ động hay bị động). Nếu kiểu liên kết là hai ngôi, thì ở hai đầu mút các đường nối, sát với các kiểu thực thể, ta ghi thêm ứng số (nếu thấy cần làm rõ).



Hình IV.7 Biểu diễn đồ họa của một kiểu liên kết

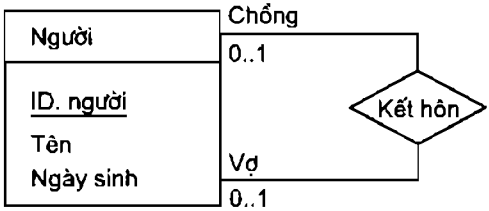
Ghi chú:

- Vị trí đặt ứng số (ở đầu nào) mà ta dùng ở đây là phù hợp với thông lệ trong các mô hình đối tượng dùng phổ biến hiện nay. Trong một số sách cũ về mô hình E/A, vị trí đặt ứng số có thể là đảo đầu so với ở đây.

- Trong phương pháp MERISE thì kiểu liên kết được biểu diễn bởi một hình elíp thay vì hình thoi.

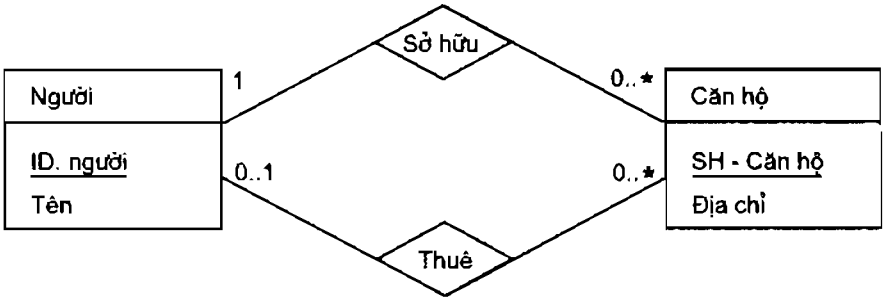
Sau đây, ta xét thêm một số trường hợp đặc biệt:

- Liên kết đệ quy: Đó là kiểu liên kết giữa một kiểu thực thể với chính nó, tức là kết nối các cặp phần tử cùng trong một kiểu thực thể. Trong trường hợp này nên ghi rõ *vai trò* của mỗi thực thể tham gia ở mỗi đầu của kiểu liên kết (Hình IV. 8).



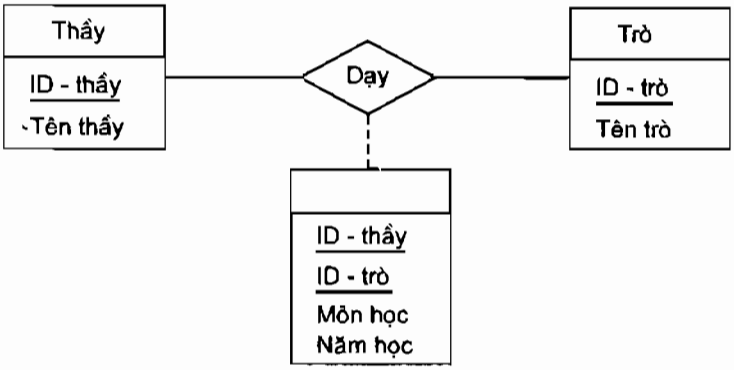
Hình IV. 8 Kiểu liên kết đệ quy

- Nhiều kiểu liên kết giữa hai kiểu thực thể: Phải vẽ chúng riêng rẽ (không được chập vào nhau) (Hình IV.9)



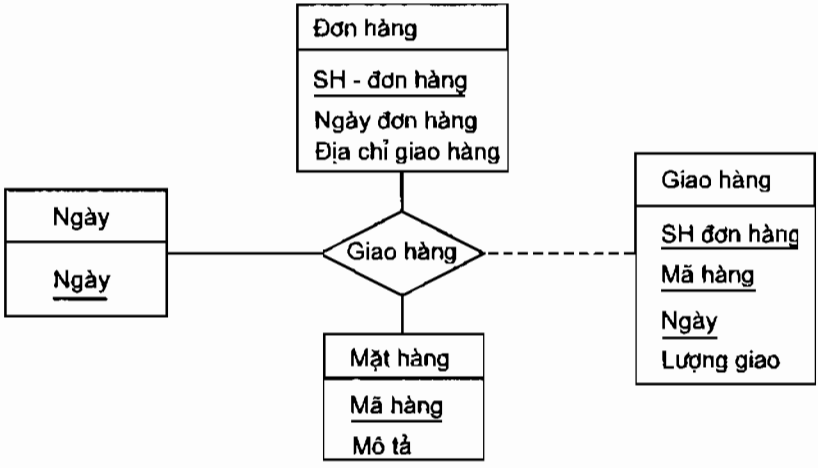
Hình IV. 9 Nhiều kiểu liên kết giữa hai kiểu thực thể.

- Kiểu liên kết có thuộc tính: Ghi danh sách các thuộc tính bên cạnh hình thoi. Cũng có thể vẽ thêm một hình chữ nhật hai ngăn như một kiểu thực thể, ngăn tên có thể chứa tên kiểu liên kết hay bỏ trống, ngăn thuộc tính chứa danh sách các thuộc tính, bổ sung các thuộc tính khóa của các kiểu thuộc tính tham gia liên kết. Hình chữ nhật này được nối với hình thoi của kiểu liên kết bằng một đường đứt nét. Cách làm này gọi là *thực thể hóa* một kiểu liên kết (Hình IV.10).



Hình IV.10 Kiểu liên kết có thuộc tính

- Kiểu liên kết nhiều ngôi: Ít gặp hơn, nhưng cũng khó quan niệm hơn. Chẳng hạn như ta biết thì một thời khóa biểu là một bảng gồm nhiều ô, mỗi ô là một bộ bốn: Môn học, Giờ học, Phòng học và Lớp sinh viên. Vậy thời khóa biểu chính là một kiểu liên kết bốn bên giữa các kiểu thực thể: Môn, Giờ, Phòng, Lớp. Cách biểu diễn vẫn là dùng hình thoi với các đường nối đến các kiểu thực thể liên quan. Nếu kiểu liên kết nhiều ngôi có thêm các thuộc tính mô tả, thì ta cũng có thể thực thể hóa như trên (Hình IV. 11).



Hình IV.11 Kiểu liên kết ba ngôi có thuộc tính mô tả

2. Mô hình thực thể/ liên kết mở rộng

Mặc dù mô hình E/A kinh điển đã được sử dụng trong các xí nghiệp và được giảng dạy trong các trường đại học khá rộng rãi, song dần dà người ta nghiệm thấy các hạn chế của nó (đặc biệt là các ràng buộc (i) và (ii) đối với các kiểu thuộc tính) làm cho nó khó thích ứng với các hệ thống phức tạp. Do đó từ sau năm 1980, người ta đã nghiên cứu đưa thêm cho nó một số điểm mở rộng. Các điểm mở rộng này chịu ảnh hưởng từ xu hướng hiện đại của mô hình hóa hướng đối tượng, cũng như của các hệ QT CSDL hướng đối tượng.

a) Các điểm mở rộng đối với mô hình E/A

Có ba điểm mở rộng:

(1) *Các kiểu thuộc tính đa trị*: Nếu như trong mô hình E/A kinh điển, do ràng buộc (i), chỉ được dùng các kiểu thuộc tính đơn trị, thì trong mô hình E/A mở rộng được phép dùng kiểu thuộc tính đa trị, nghĩa là kiểu thuộc tính mà giá trị của nó đối với một thực thể có thể là 1 dãy hay 1 tập các giá trị đơn.

Thí dụ: - Tên các con, Tuổi các con của một nhân viên

- Các số điện thoại của một đơn vị

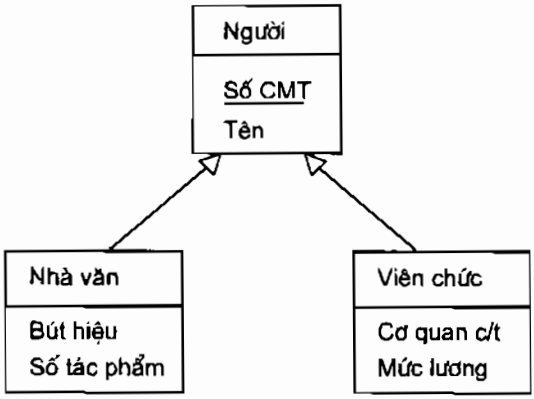
(2) *Các kiểu thuộc tính phức hợp*: Trong mô hình E/A kinh điển, do ràng buộc (ii), không được dùng các kiểu thuộc tính là tổ hợp hay hạn chế từ nhiều kiểu thuộc tính khác. Hướng mở rộng ở đây cho phép dùng các kiểu thuộc tính đó (gọi là kiểu thuộc tính phức hợp), tạo thành bởi sự kết tập từ nhiều kiểu thuộc tính khác. Một cách mặc định, thì mỗi giá trị của kiểu thuộc tính phức hợp là sự ghép tiếp các giá trị của các kiểu thuộc tính sơ đẳng.

Thí dụ: Kiểu thuộc tính Địa chỉ là sự kết tập các kiểu thuộc tính: Số nhà, Đường phố, Quận Huyện, Tỉnh Thành.

(3) *Các kiểu thực thể con*: Xuất hiện bởi yêu cầu khái quát hóa hay chuyên biệt hóa khi cần phân cấp các sự vật. Nếu trong một kiểu thực thể A, ta chỉ ra một tập con B của A, mà các thực thể trong B vừa mang các kiểu thuộc tính chung của các thực thể trong A, lại vừa có thêm một số các kiểu thuộc tính mới, thì ta nói đó là sự chuyên biệt hóa. B được gọi là *kiểu thực thể con* của kiểu thực thể A. Các kiểu thuộc tính của B bao gồm mọi kiểu thuộc tính của A cộng thêm các kiểu thuộc tính riêng của nó. Ta nói: B *thừa kế* các thuộc tính của A. Quá trình ngược lại với chuyên biệt hóa là sự khái quát hóa: Từ nhiều kiểu thực thể B, C,... ta rút ra các kiểu thuộc tính chung để lập một kiểu thực thể A (với các kiểu thuộc tính chung đó) sao cho B, C,... đều là kiểu thực thể con của A.

Nếu B là kiểu thực thể con của kiểu thực thể A, thì trong biểu diễn đồ họa, ta vẽ một mũi tên (đầu tam giác) từ B tới A.

Thí dụ: Nhà văn và Viên chức đều là các kiểu thực thể con của Kiểu thực thể Người (Hình IV.12).



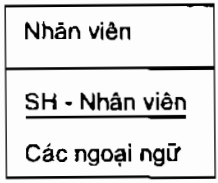
Hình IV.12 Các kiểu thực thể con

b) Cách biến đổi một biểu đồ E/A mở rộng về một biểu đồ E/A kinh điển

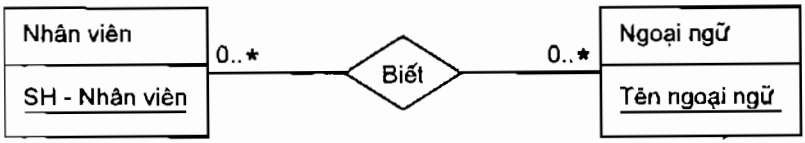
Các hệ QT CSDL quan hệ ngày nay đòi hỏi giá trị đơn cho các kiểu thuộc tính và cấm khái niệm thừa kế giữa các quan hệ. Do đó để thành lập một CSDL quan hệ bắt đầu từ một lược đồ khái niệm theo mô hình E/A mở rộng, thì trước hết cần phải chuyển lược đồ này sang mô hình E/A kinh điển đã. Sau đây là các quy tắc để biến đổi mô hình E/A mở rộng sang mô hình E/A kinh điển.

Quy tắc 1: Xử lý các thuộc tính đa trị của một kiểu thực thể: Thay một kiểu thuộc tính đa trị T của một kiểu thực thể A bởi một kiểu thực thể mới E-T và kết nối A với E-T bởi một kiểu liên kết. Đưa vào kiểu thực thể mới E-T một kiểu thuộc tính đơn trị t, tương ứng với giá trị thành phần của T. Nghiên cứu ứng số cho kiểu liên kết mới (giữa A và E-T).

Thí dụ: - Kiểu thực thể nhân viên có kiểu thuộc tính đa trị là: các ngoại ngữ (các nhân viên kẻ ít người nhiều biết một số ngoại ngữ):



- Chuyển sang mô hình E/A kinh điển, ta sẽ có:



Hình IV.13 Xử lý kiểu thuộc tính đa trị

Ghi chú:

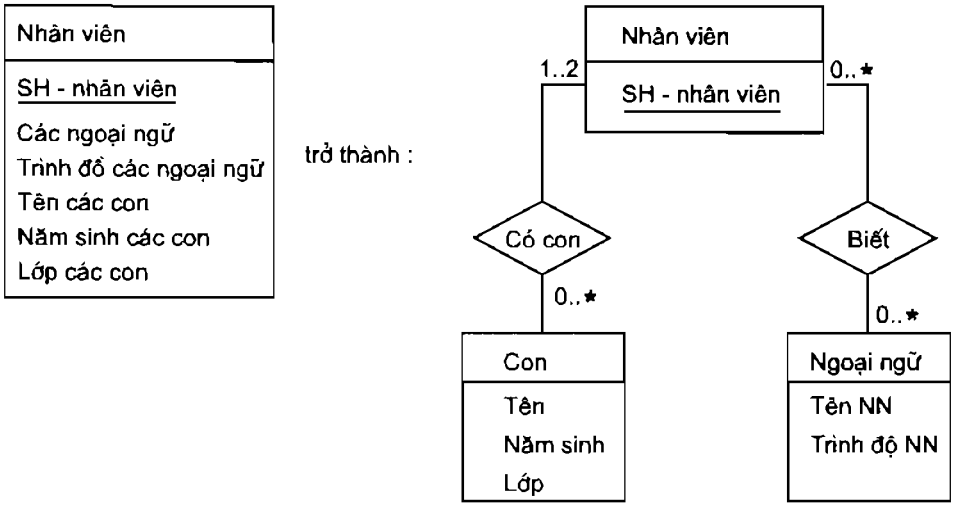
- Kiểu thực thể mới E-T nói trên thường được gọi là *kiểu thực thể phụ thuộc*. Kiểu thực thể phụ thuộc chỉ tồn tại cùng với kiểu thực thể chính. Nghĩa là khi kiểu thực thể chính vì một lý do nào đó không còn nữa, thì kiểu thực thể phụ thuộc nó cũng phải bị loại bỏ.

- Nếu kiểu thuộc tính đa trị T có giá trị luôn luôn gồm một số lượng nhất định n các giá trị đơn, thì không cần đưa thêm kiểu thực thể phụ thuộc, mà chỉ việc thay T bởi n kiểu thuộc tính đơn trị T_1, T_2, ..., T_n Chẳng hạn, nếu ta quy định rằng mỗi nhân viên chỉ cần kê khai hai ngoại ngữ (giỏi nhất) thì lúc đó ta chỉ cần biến đổi như trên Hình IV.14.

Hình IV.14 Xử lý kiểu thuộc tính đa trị có độ dài cố định

- Nếu kiểu thực thể A chứa một cụm các kiểu thuộc tính đa trị cùng diễn tả về một chủ đề chung (ví dụ: Các ngoại ngữ, trình độ các ngoại ngữ hoặc là

Tên các con, Năm sinh các con, Lớp các con), thì cả cụm thuộc tính liên quan với nhau đó được chuyển thành một kiểu thực thể phụ thuộc bao gồm các kiểu thuộc tính đơn trị tương ứng, chứ không phải thành nhiều kiểu thực thể (Hình IV.15).



Hình IV.15 Cụm các kiểu thuộc tính đa trị theo chủ đề

- Một trường hợp đặc biệt quan trọng thường gặp trong mô hình hóa là trường hợp các *chứng từ có bảng*. Chẳng hạn một Hóa đơn như ở Hình IV.16. Trong mẫu Hóa đơn đó ta thấy có nhiều *tiêu thức*, tức là tên của các thông tin mà người lập hóa đơn phải điền giá trị vào chỗ trống ngay sau đó (hay dưới đó). Tuy nhiên, ta nhận thấy có hai loại tiêu thức khác nhau: các tiêu thức ở ngoài bảng, chỉ nhận một giá trị duy nhất, và các tiêu thức trong bảng (viết đầu các cột) thì lại có thể nhận nhiều giá trị. Khi tiến hành mô hình hóa, một cách tự nhiên ta diễn tả Hóa đơn là một kiểu thực thể, với tất cả các tiêu thức lấy làm kiểu thuộc tính. Như thế ta có một mô hình E/A mở rộng (Hình IV.17). Sau đó ta biến đổi mô hình E/A mở rộng đó thành mô hình E/A kinh điển bằng cách loại bỏ các thuộc tính đa trị (tương ứng với các tiêu thức nằm trong bảng). Ở đây cũng chỉ cần tạo ra một kiểu thực thể phụ thuộc duy nhất, mà mỗi thực thể trong đó tương ứng với một *dòng* các giá trị viết trong bảng (Hình IV.18).

HOÁ ĐƠN

Số .....

Họ tên người mua .....

Địa chỉ .....

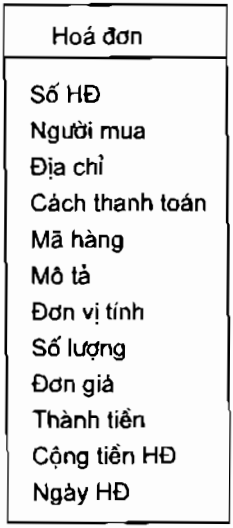
Hình thức thanh toán .....

Mã hàng	Tên, quy cách	Đơn vị tính	Số lượng	Đơn giá	Thành tiền

Cộng tiền:.....

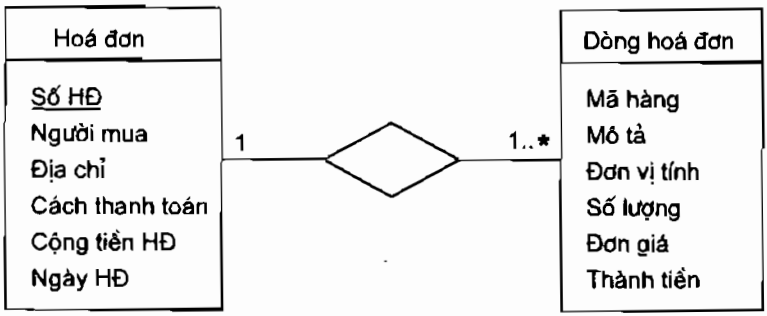
Ngày.....

Hình IV.16 Một chứng từ có bảng



Hình IV.17 Kiểu thực thể Hóa đơn
(Mô hình E/A mở rộng)

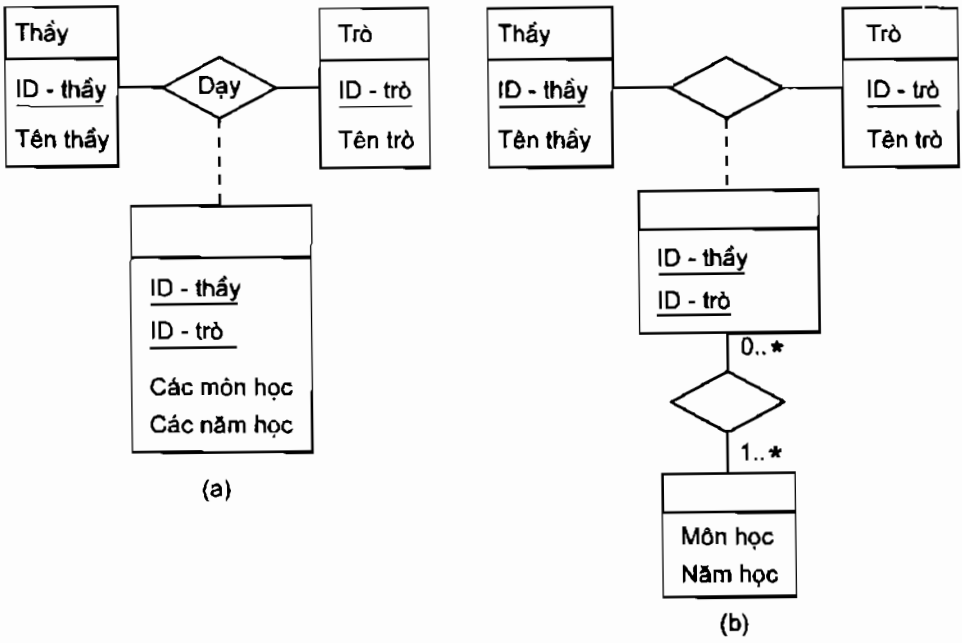
Ghi chú: Tên một số tiêu thức đã được sửa lại để cho gọn hơn và để tránh nhầm lẫn.



Hình IV.18 Diễn tả một chứng từ có bảng trong mô hình E/A kinh điển (bằng 2 kiểu thực thể chính và phụ thuộc).

Quy tắc 2: Xử lý các kiểu thuộc tính đa trị của một kiểu liên kết: Thực thể hóa kiểu liên kết đó (xem mục 1.b) rồi áp dụng quy tắc 1 cho kiểu thực thể mới lập.

Thí dụ: Trở lại thí dụ cho ở Hình IV.10, nhưng giả sử rằng kiểu liên kết Dạy giữa hai kiểu thực thể Thầy và Trò giờ đây có hai kiểu thuộc tính đa trị là Các môn học và Các năm học, thì ta có phép biến đổi như trong Hình IV.19.



Hình IV.19 (a) Mô hình E/A mở rộng
(b) Mô hình E/A kinh điển tương đương

Quy tắc 3: Xử lý các kiểu thuộc tính phức hợp: Thay kiểu thuộc tính phức hợp bởi các kiểu thuộc tính hợp thành.

Thí dụ: Thay Địa chỉ bởi Số nhà, Đường phố, Quận huyện và Tỉnh thành.

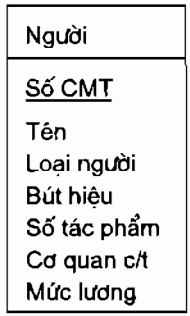
Quy tắc 4: Xử lý các kiểu thực thể con:

Giả sử kiểu thực thể A có kiểu thực thể con là B. Có hai cách xử lý tùy chọn như sau:

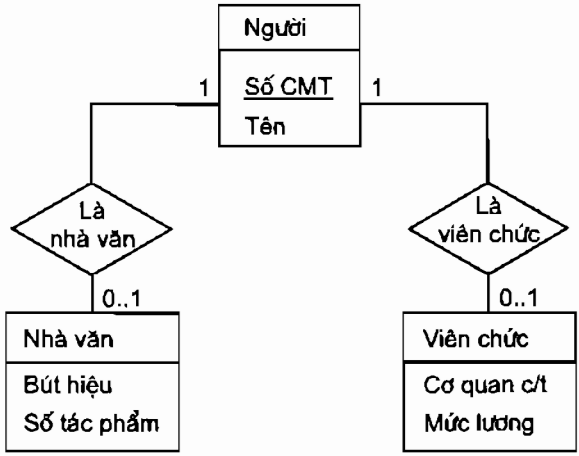
Quy tắc 4.1: Loại bỏ kiểu thực thể B và bổ sung mọi kiểu thuộc tính của B vào trong A, đồng thời thêm một kiểu thuộc tính cho phép phân loại các thực thể của A (thuộc B hay không thuộc B). Chuyển mọi kiểu liên kết với B sang A, và nghiên cứu lại các ứng số cho chúng.

Quy tắc 4.2: Thay mối liên quan thừa kế giữa A và B bởi một kiểu liên kết giữa A và B mà các ứng số tối đa đều là 1. Nghiên cứu cụ thể các ứng số tối thiểu.

Thí dụ: Trở lại thí dụ trong Hình IV.12. Áp dụng quy tắc 4.1 ta có kiểu thực thể như trong Hình IV.20, còn áp dụng quy tắc 4.2 ta lại có biểu đồ như trong Hình IV.21.



Hình IV.20 Thay các kiểu thực thể con bằng một kiểu thuộc tính phân loại.

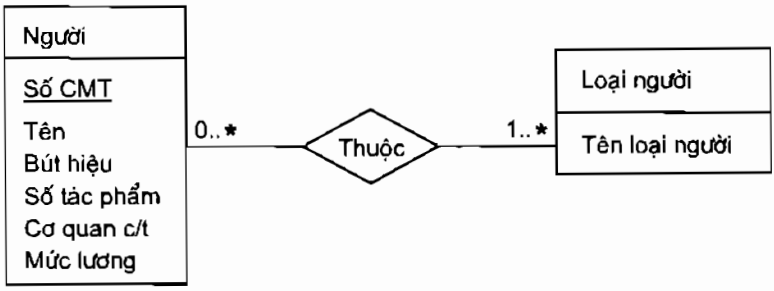


Hình IV.21 Thay quan hệ thừa kế bằng một kiểu liên kết

Chú thích:

- Trong hình IV.20, khi Loại người = Viên chức, thì các kiểu thuộc tính Bút hiệu và Số tác phẩm không dùng tới, nghĩa là không có giá trị. Tuy nhiên để cho các kiểu thuộc tính đối với một thực thể luôn luôn có giá trị, trong trường hợp như trên, người ta gán cho kiểu thuộc tính một giá trị quy ước: Null (được hiểu là không tồn tại hoặc chưa được biết).

- Trong Hình IV.20, nếu kiểu thuộc tính Loại người lại là thuộc tính đa trị (nghĩa là có thể có người vừa là nhà văn vừa là viên chức), thì lại phải vận dụng quy tắc 1 để xử lý tiếp, mà kết quả sẽ như trong hình IV.22.



Hình IV.22 Thay các kiểu thực thể con bằng một kiểu thực thể trỏ loại.

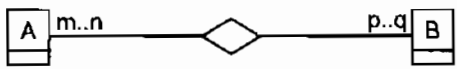
3. Mô hình thực thể/ liên kết hạn chế

Mô hình thực thể/ liên kết hạn chế mà ta đề cập sau đây, tuy bị hạn chế nhiều về các hình thức diễn tả (do đó vận dụng khó hơn), nhưng lại rất gần với mô hình quan hệ và do đó lại dễ dàng chuyển sang cài đặt với một hệ QT CSDL quan hệ hơn. Đây chính là mô hình được dùng trong một số hệ trợ giúp thiết kế (CASE), chẳng hạn trong ORACLE.

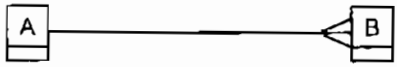
a) Các hạn chế

So với mô hình E/A kinh điển, thì ngoài các ràng buộc (i) và (ii), ở đây có thêm các hạn chế mới như sau:

• Đối với các kiểu liên kết hai ngôi dạng:



thì chỉ còn được dùng kiểu liên kết 1 - nhiều, tức là trường hợp n=1 và q>1 hoặc q=* và được biểu diễn dưới dạng mới như sau:

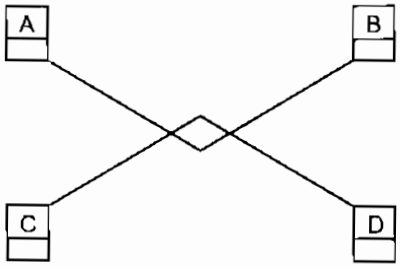


Đặc biệt trường hợp có thêm m = 0, ta gọi đó là kiểu liên kết 0/1 - nhiều, và biểu diễn như sau:

Tuy nhiên cách biểu diễn này cũng ít khi được dùng (nghĩa là không cần phân biệt rõ 0/1 - nhiều với 1 - nhiều).

Như vậy, các kiểu liên kết 1-1 (n=1và q=1), cũng như các kiểu liên kết nhiều - nhiều (n >1 và q>1) đều không được biểu diễn tường minh trong mô hình E/A hạn chế. Chính vì hạn chế này mà mô hình E/A hạn chế đôi khi còn được gọi là *mô hình E/A một - nhiều.*

- Đối với các kiểu liên kết nhiều ngôi (số ngôi ≥ 3) dạng:



thì cũng không còn hình thức diễn tả tường minh trong mô hình E/A hạn chế.

Với hai hạn chế như trên, thì rốt cục mô hình E/A hạn chế chỉ còn là một tập hợp các kiểu thực thể (kèm danh sách các kiểu thuộc tính) kết nối với nhau bởi các kiểu liên kết 1- nhiều (tức là các đường nối có xòe chân vịt ở một đầu và không mang tên).

b) Cách biến đổi một mô hình E/A kinh điển về một mô hình E/A hạn chế

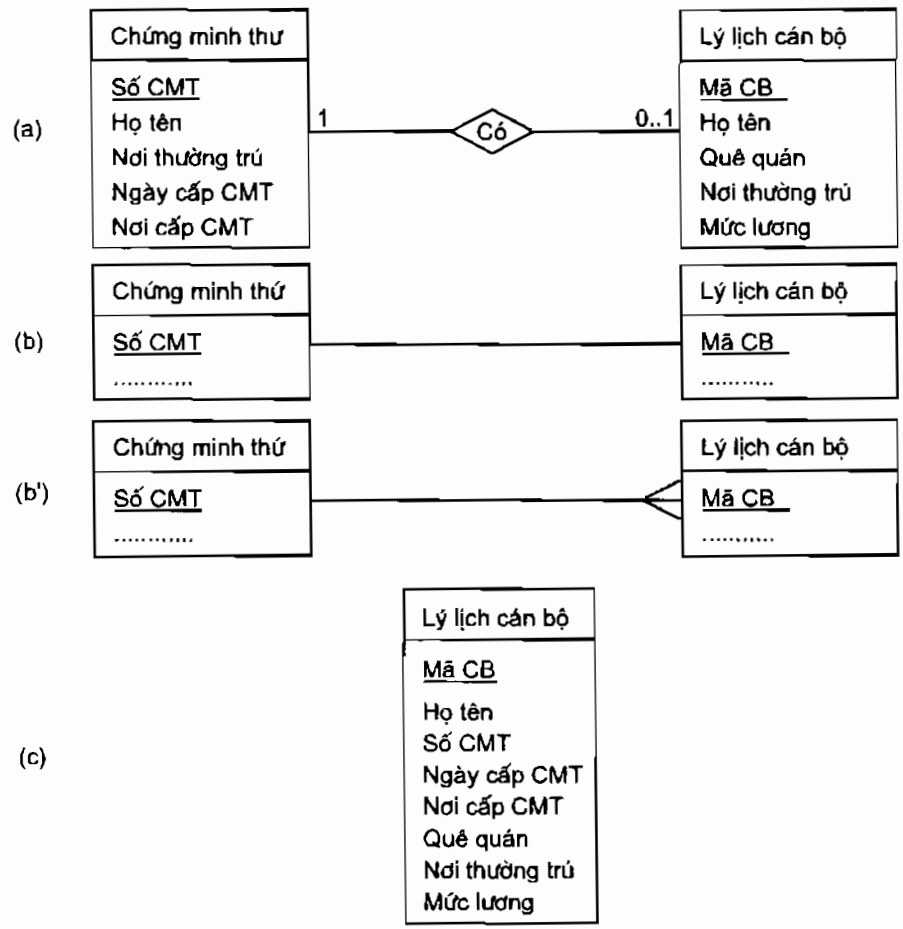
Tuy bị hạn chế rất ngặt nghèo về hình thức biểu diễn, song khả năng diễn tả của mô hình E/A hạn chế vẫn không hề bị giảm sút. Bằng cứ là, ta có thể biến đổi mọi mô hình E/A kinh điển về mô hình E/A hạn chế nhờ một số quy tắc như sau:

Quy tắc 5: Xử lý các kiểu liên kết 1-1.

Tùy ý thực hiện theo hai cách:

- *Cách 1:* Xem 1-1 là trường hợp riêng của 1-nhiều (bởi vì "nhiều" có nghĩa là p..*, vậy khi p=0 sẽ được hiểu là 0,1 hay nhiều) và vẽ lại nó bằng một đường nối thẳng, hay một đường nối có chân vịt ở một đầu. Cách làm này vi phạm sự hạn chế của mô hình, hoặc dễ gây hiểu lầm, nên ít dùng.
- *Cách 2*: Gộp hai kiểu thực thể có quan hệ 1-1 thành một kiểu thực thể duy nhất, bằng cách hòa trộn hai danh sách các kiểu thuộc tính với nhau.

Thí dụ: Hai kiểu thực thể Chứng minh thư và Lý lịch cán bộ có quan hệ 1-1. Có thể vẫn để nguyên hoặc gộp vào nhau như trong hình IV.23.

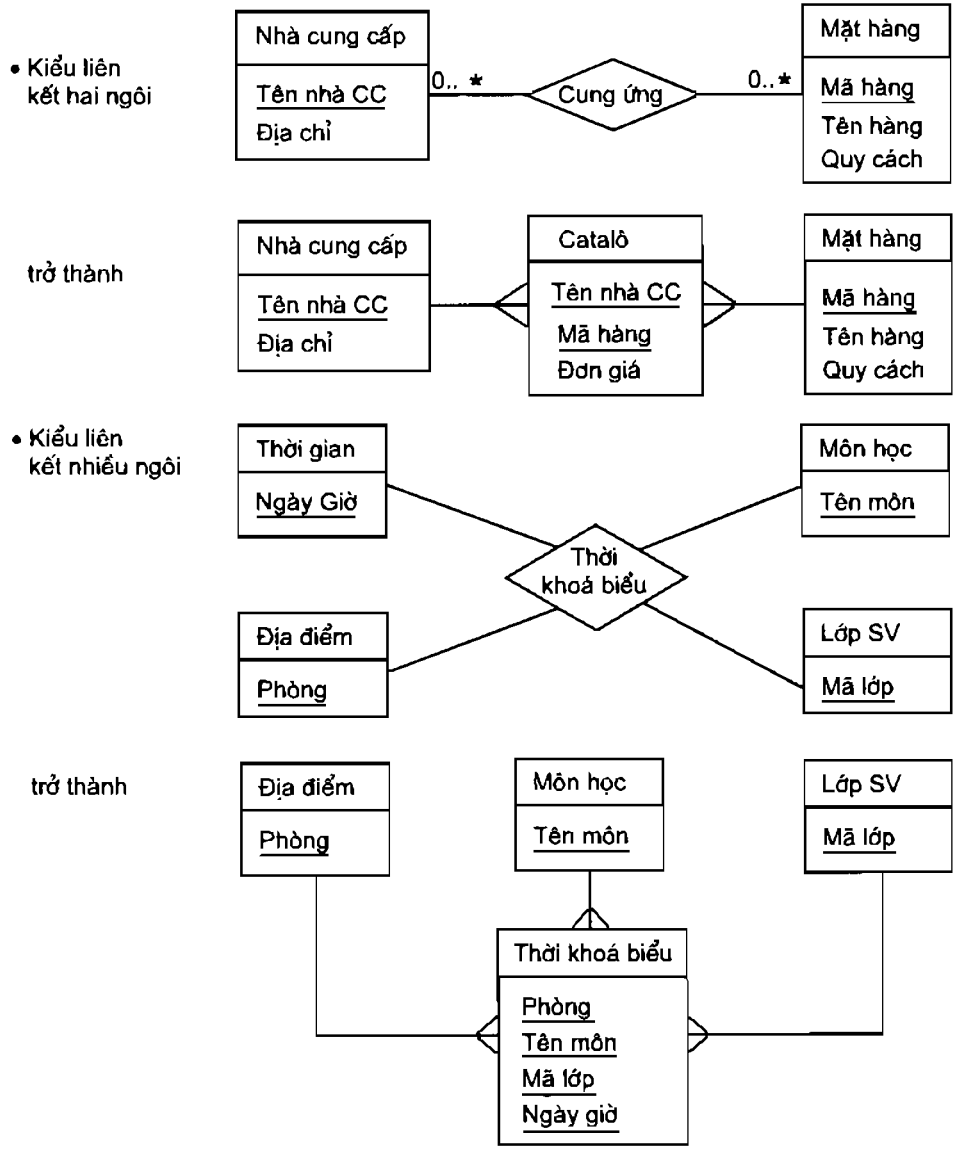


Hình IV. 23 Xử lý các kiểu liên kết 1-1 :

(a) Mô hình E/A kinh điển; (b), (b') Biến đổi theo cách 1; (c) Biến đổi theo cách 2.

Quy tắc 6: Xử lý các kiểu liên kết 2 ngôi nhiều - nhiều và các kiểu liên kết nhiều ngôi: Thực thể hóa mỗi kiểu liên kết đó bằng một kiểu thực thể mới có chứa các kiểu thuộc tính là khóa của các kiểu thực thể tham gia (tập hợp các khóa này tạo thành các khóa bội của kiểu thực thể mới). Nối kiểu thực thể này với các kiểu thực thể tham gia liên kết bằng các liên kết 1 - nhiều (các chân vịt vẽ bám vào kiểu thực thể mới).

Thí dụ:



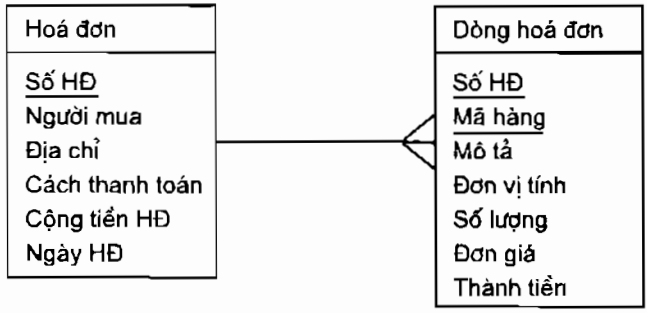
Hình IV.24 Thực thể hóa các kiểu liên kết nhiều - nhiều hay nhiều ngôi.

Ghi chú: Thông thường thì thời gian ít được xem là một kiểu thực thể, mà chỉ là một kiểu thuộc tính (như là một loại tham số).

c) Các kiểu thuộc tính khóa và kiểu thuộc tính kết nối

Đối với mô hình E/A hạn chế, ta cần phải chỉ rõ khóa cho mỗi kiểu thực thể. Khóa đó có thể là khóa đơn (chỉ gồm một kiểu thuộc tính), hoặc là khóa bội (gồm nhiều kiểu thuộc tính). Khóa bội thường gặp ở các trường hợp:

- Các kiểu thực thể phụ thuộc: Khóa của một kiểu thực thể phụ thuộc luôn phải bao gồm khóa của kiểu thực thể chính như một thành phần của khóa, và nó làm nhiệm vụ kết nối một thực thể phụ thuộc với một thực thể chính (Hình IV.25).
- Các kiểu thực thể lập từ Quy tắc 6, tức là kiểu thực thể diễn tả một quan hệ (một kiểu liên kết nhiều - nhiều hay nhiều ngôi) đều phải có khóa bội hợp thành từ khóa của các kiểu thực thể tham gia quan hệ (Hình IV.24).

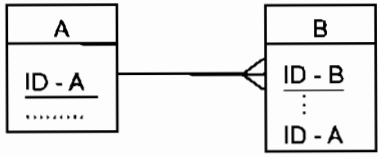


Hình IV. 25 Các kiểu thực thể chính và phụ thuộc

Ta gọi một *kiểu thuộc tính kết nối* là một kiểu thuộc tính vốn là khóa của một kiểu thực thể, nhưng lại xuất hiện trong một kiểu thực thể khác với nhiệm vụ mô tả mối quan hệ giữa hai kiểu thực thể. Kiểu thuộc tính kết nối còn có tên khác là *khóa ngoài*.

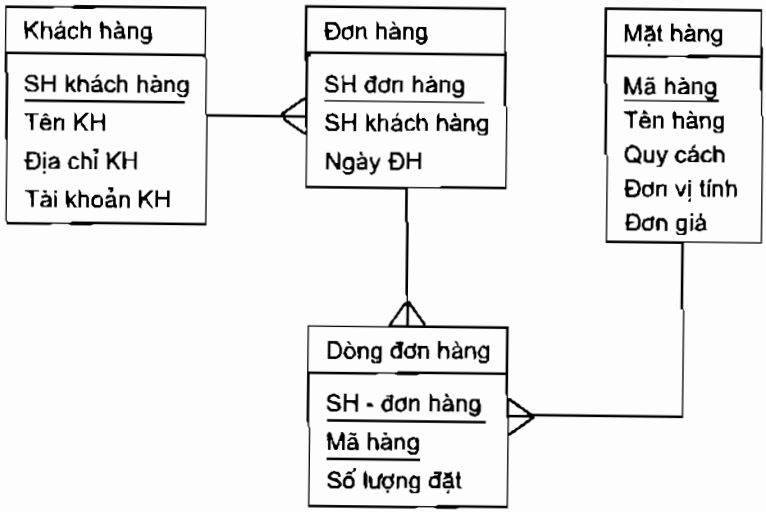
Dễ dàng rút ra quy luật sau:

Nếu một kiểu thực thể B có chứa một kiểu thuộc tính kết nối, là khóa của một kiểu thuộc tính A, thì giữa A và B có một kiểu liên kết 1 - nhiều (đầu nhiều về phía B) (Hình IV.26).



Hình IV. 26 Phản ánh kiểu liên kết 1 - nhiều bởi thuộc tính kết nối

Thí dụ: Biểu đồ E/A hạn chế cho trong Hình IV.27 phản ánh sự liên quan chặt chẽ giữa các kiểu liên kết 1 - nhiều với các thuộc tính kết nối.



Hình IV.27 Các thuộc tính kết nối phản ánh các kiểu liên kết 1 - nhiều

Ghi chú: Nếu trong một mô hình E/A hạn chế, ta sử dụng một cách có hệ thống các thuộc tính kết nối để phản ánh các kiểu liên kết 1 - nhiều giữa các kiểu thực thể, thì các đường nối 1 - nhiều trở nên thừa, có thể xóa bỏ. Lúc đó mô hình chỉ còn là một tập hợp các kiểu thực thể. Vì thế mô hình E/A hạn chế còn có tên gọi khác là *mô hình thực thể* (Entity model). Tuy nhiên, dù là thừa, ta vẫn giữ lại các đường nối 1 - nhiều trong mô hình, nhằm làm cho mô hình dễ đọc, dễ hiểu hơn.

4. Phương pháp phân tích dữ liệu theo mô hình E/A

Việc phân tích dữ liệu thường thực hiện theo hai giai đoạn:

- Đầu tiên lập lược đồ dữ liệu theo mô hình thực thể/liên kết, nhằm phát huy thế mạnh về tính trực quan và dễ vận dụng của mô hình này.
- Tiếp đó hoàn thiện lược đồ dữ liệu theo mô hình quan hệ, nhằm lợi dụng cơ sở lý luận chặt chẽ của mô hình này trong việc chuẩn hóa lược đồ.

Mô hình quan hệ sẽ được đề cập ở Chương V tiếp sau, còn ở đây ta chỉ đề cập việc lập lược đồ dữ liệu theo mô hình thực thể/liên kết.

a) Mục đích và yêu cầu của việc phân tích dữ liệu

Mục đích phân tích dữ liệu của hệ thống là lập lược đồ khái niệm về dữ liệu, làm căn cứ cho việc thiết kế cơ sở dữ liệu của hệ thống sau này. Lược đồ khái niệm về dữ liệu ở giai đoạn đầu được lập dưới dạng mô hình E/A. Như ta đã nghiên cứu ở trên, có ba mức của mô hình E/A:

- mức mở rộng,
- mức kinh điển,
- mức hạn chế

Có thể lập lược đồ dữ liệu theo mức nào cũng được, vì từ cả ba loại mô hình E/A này ta đều có thể chuyển sang mô hình quan hệ để tiếp tục công việc của giai đoạn hai. Song tốt nhất là nên bắt đầu với mô hình E/A mở rộng, vì mô hình này cho phép dùng các kiểu thuộc tính đa trị hay phức hợp, lại có sẵn hình thức để diễn tả quan hệ kế thừa, cho nên dễ thích ứng với thực tế hơn. Tiếp đó ta biến đổi mô hình E/A mở rộng sang mô hình E/A kinh điển, rồi sang mô hình E/A hạn chế (vận dụng các quy tắc 1-6). Sở dĩ ta biến đổi về mô hình E/A hạn chế, vì mô hình này khá gần với cái đích mà ta muốn đạt đến hơn (gần với dạng chuẩn 3 của mô hình quan hệ). Đương nhiên, với những người phân tích có kinh nghiệm, thì có thể lập ngay lược đồ dữ liệu theo mô hình E/A hạn chế, mà bỏ qua quá trình đi qua các mô hình E/A mở rộng và kinh điển.

Hai yêu cầu đối với việc phân tích dữ liệu là:

- *Không bỏ sót thông tin*: nghĩa là phải phát hiện để đưa vào lược đồ dữ liệu mọi thông tin cần phải được thu thập lưu giữ, đủ để phục vụ cho các xử lý trong hệ thống.
- *Không dư thừa thông tin*: nghĩa là các thông tin đưa vào lược đồ không được trùng lắp (cùng một thông tin được ghi nhận nhiều lần, ở nhiều nơi), và không được là thông tin có thể suy dẫn từ các thông tin khác đã có. Sự dư thừa không những làm tốn chỗ nhớ, nhưng quan trọng hơn là dễ gây ra sự mâu thuẫn (sự không nhất quán về giá trị) của thông tin lưu giữ trong cơ sở dữ liệu.

Để khỏi bỏ sót thông tin, ta phải rà soát kỹ để tìm ra chúng từ nhiều nguồn điều tra có thể, như là:

- Các cuộc tiếp xúc, phỏng vấn với các nhân viên (kể cả với ban lãnh đạo);
- Các loại tư liệu về hệ thống: đó là các báo cáo, các thuyết minh, các tài liệu lập trong các bước khảo sát hiện trạng và phân tích chức năng trước đó. Trong các tư liệu đó, thì mỗi danh từ đều nên được cân nhắc xem có

đáng là một thông tin đưa vào lược đồ không (là một thuộc tính hay một thực thể). Ví dụ các danh từ được gạch dưới trong câu sau, trích trong một tư liệu, là đáng lưu ý:

> "... Công ty chúng tôi hiện tại đã phát triển tới mức sử dụng 5.000 nhân viên và phục vụ cho các khách hàng rải ra trên khắp nước và cả ở nước ngoài...".

- Các biểu đồ luồng dữ liệu lập ở giai đoạn phân tích chức năng.

Trong biểu đồ luồng dữ liệu, các luồng dữ liệu và nhất là các kho dữ liệu sẽ gợi ý cho ta các thông tin cần đưa vào cơ sở dữ liệu của hệ thống.

- Các loại sổ sách, các tệp: đó là các thông tin có cấu trúc đã được lưu trong hệ thống cũ, dùng cho xử lý thủ công hay xử lý máy tính, có nhiều khả năng cũng rất cần cho hệ thống mới.

b) Hai cách tiến hành: trên xuống và dưới lên

Các thông tin được đưa vào mô hình E/A dưới ba hình thức: thực thể, liên kết và thuộc tính. Các liên kết thì ta sẽ đề cập sau ở phần dưới. Trước hết hãy đề cập việc phát hiện các kiểu thực thể và các kiểu thuộc tính của chúng.

Có thể tiến hành theo hai cách ngược nhau:

Cách (1): Trước hết đối sánh với thực tế để tìm ra các kiểu thực thể, rồi sau đó đối với mỗi kiểu thực thể đã chọn, tìm các kiểu thuộc tính mô tả nó.

Cách (2): Tìm tập hợp các kiểu thuộc tính như là những loại thông tin sơ đẳng xuất hiện trong hệ thống, rồi gom nhóm, tổ chức chúng lại thành từng cụm theo chủ đề mô tả. Mỗi cụm đó trở thành một kiểu thực thể (hay cũng có thể là một kiểu liên kết).

Tuy nhiên cách tốt nhất là hỗn hợp cả hai cách làm trên bằng cách lặp đi lặp lại nhiều lần: Từ các kiểu thực thể, tìm ra các kiểu thuộc tính, rồi từ các kiểu thuộc tính ta tổ chức lại, chỉnh sửa và bổ sung thành các kiểu thực thể, vòng đi vòng lại cho đến khi thấy thỏa đáng.

Một thực thể - như ta đã định nghĩa ở trên - là một vật thể cụ thể hay trừu tượng, tồn tại thực sự và khá ổn định trong thế giới thực mà ta muốn phản ánh nó trong hệ thống thông tin.

Theo Coad và Yourdon, thì thực thể có thể là:

- thực thể vật chất (cụ thể): ôtô, máy bay, cảm biến...

- vai trò: sinh viên, người cung cấp, khách hàng,...

- sự kiện: đặt hàng, đăng ký xe máy, mượn sách...
- địa điểm: văn phòng, cửa hàng...
- tổ chức: phòng ban, khoa, lớp,...

v.v...

Tuy nhiên, không phải mọi vật thể có trong thực tế đều trở thành thực thể, mà phải có sự tuyển chọn khi đưa vào mô hình. Sự tuyển chọn đó dựa trên hai tiêu chí:

- có ích cho việc quản lý,
- có cách để phân biệt với các thực thể khác.

Chẳng hạn, trong một nhà máy sản xuất đanh, thì từng chiếc đanh đều là vật thể rất cụ thể, song ta lại không chọn các chiếc đanh làm thực thể. Sở dĩ thế không những là vì thật khó (hoặc quá cầu kỳ) để đưa ra cách phân biệt các chiếc đanh với nhau, mà chủ yếu là vì theo dõi thông tin về từng chiếc đanh thì cũng chẳng mang lại lợi ích gì cho việc quản lý. Trái lại, các đối tượng trừu tượng hơn là các loại đanh, như Đanh 10 phân, Đanh đóng guốc, Đanh bêtông... lại đáng được chọn làm các thực thể trong mô hình (chúng thuộc cùng một kiểu thực thể là loại đanh mà các kiểu thuộc tính mô tả có thể là: Kích cỡ, Chất liệu, Trọng lượng đóng gói, Đơn giá xuất xưởng, v.v...).

Một thuộc tính - như đã định nghĩa ở trên - là một giá trị (dữ liệu) mô tả một khía cạnh nào đó của một thực thể (hay của một liên kết).

Cũng không phải mọi đặc điểm có thể của một thực thể đều được xem là thuộc tính cho nó, mà phải có sự tuyển chọn theo yêu cầu quản lý.

Xét về vai trò của kiểu thuộc tính trong một kiểu thực thể, ta phân biệt ba loại kiểu thuộc tính:

- Thuộc tính khóa,
- Thuộc tính kết nối (hay khóa ngoài),
- Thuộc tính mô tả.

Về các thuộc tính khóa và kết nối, ta sẽ đề cập ở dưới cùng với việc xem xét các liên kết. Còn đối với các thuộc tính mô tả, để tuyển chọn chúng, ta dựa vào một số cách phân loại, đề cập ở hai mục nhỏ tiếp sau đây.

c) Phân loại các thuộc tính theo nội dung diễn tả

Như đã nói ở Chương I, các dữ liệu trong một hệ thống thông tin quản lý (nói ở đây là các thuộc tính và từ đó là các thực thể) có thể tách làm hai loại theo nội dung diễn tả:

- Các dữ liệu phản ánh cấu trúc tĩnh của doanh nghiệp. Có thể nói đó là phần mô tả hạ tầng cơ sở của doanh nghiệp và môi trường của nó, và là phần ổn định nhất trong CSDL, bao gồm:

  + Về nhân lực: Nhân viên, Khách hàng, nhà cung cấp...

  + Về tài sản, thiết bị, nguyên vật liệu: Nhà xưởng, Cửa hàng, Máy, Ô tô, Mặt hàng, Nguyên liệu, ...

  + Về tổ chức và cơ cấu: Phòng ban, Phân xưởng, Khoa, Bộ môn, Lớp SV, ...

- Các dữ liệu phản ánh các sự kiện trong hoạt động kinh doanh, như:

  + Các giao dịch: Đặt hàng, Đăng ký, Giao hàng, Thanh toán,...

  + Các báo cáo, tổng kết, thống kê.

Trên một góc nhìn khác, lại có thể phân chia các dữ liệu thành:

- Các dữ liệu sống: là các dữ liệu phản ánh thực tế hiện tại của doanh nghiệp và đang được sử dụng. Các dữ liệu này phải luôn luôn được cập nhật cho khỏi lạc hậu với thực tế.

Thí dụ: Lý lịch các nhân viên đang tại chức, Đơn hàng đang xử lý (chưa hoàn tất),...

- Các dữ liệu lịch sử: là các dữ liệu phản ánh quá khứ của doanh nghiệp, được chuyển sang lưu trữ, không cập nhật, không dùng trong các tính toán xử lý, và chỉ có thể tham khảo khi cần. Thường thì các dữ liệu lịch sử được lưu trên bộ nhớ tách biệt với các dữ liệu sống (cho nên thường được gọi là "Cơ sở ngoài luồng").

Thí dụ: Hồ sơ về các nhân viên đã ra khỏi cơ quan, Đơn hàng đã hoàn tất, Tài khoản đã đóng,...

d) Phân loại các thuộc tính theo đặc điểm về giá trị của nó

Xét về đặc điểm của giá trị mà thuộc tính có thể nhận, thì ta có thể phân loại dựa trên hai tiêu chí sau:

- Thuộc tính tính toán hay thuộc tính không tính toán;

- Thuộc tính bền vững hay thuộc tính không bền vững.

Thuộc tính tính toán (còn gọi là thuộc tính dẫn xuất) là thuộc tính mà giá trị của nó thu được bởi sự tính toán theo một công thức (hay quy trình) xuất phát từ giá trị của một số thuộc tính khác. Chẳng hạn:

Thành tiền = Đơn giá × Số lượng

Có một loại thuộc tính tính toán đặc biệt là: thuộc tính mà giá trị của nó được tính toán bằng sự tích lũy các giá trị đã qua của một vài thuộc tính khác (cho đến hiện tại). Chẳng hạn:

Số dư tiết kiệm = Tổng số tiền các lần gửi vào - Tổng số tiền các lần rút ra

Loại thuộc tính tính toán bằng sự tích lũy theo thời gian này thường phản ánh thực trạng cho đến ngày hôm nay của một thực thể nào đó qua một quá trình biến diễn trong thời gian. Bởi vậy ta thường gọi đó là các *thuộc tính tình thế*.

Thí dụ:

- Lượng hàng tồn kho (của một mặt hàng) qua nhiều lần xuất, nhập.
- Số dư tài khoản (của một khách hàng) qua nhiều lần giao dịch nợ/có.

Nhớ rằng hai yêu cầu đối với lược đồ khái niệm về dữ liệu là các thông tin phải đủ và không dư thừa. Thế thì các thuộc tính tính toán là dư thừa, vậy về nguyên tắc là không được đưa vào lược đồ. Tuy nhiên các thuộc tính tính toán, nhất là các thuộc tính tình thế không phải là không có ích. Sau này đến giai đoạn thiết kế, khi xét thêm hai yêu cầu mới là: tiện lợi và nhanh chóng, thì rất có thể là lại phải đưa chúng vào cơ sở dữ liệu. Vậy ở giai đoạn phân tích, ta hãy ghi nhận chúng lại để sau này trở lại xem xét, thậm chí nếu thấy vai trò của chúng là không thể thiếu (ví dụ: Tổng tiền trong một hóa đơn) thì ta cứ đưa vào lược đồ, nhưng có đánh dấu đặc biệt (chẳng hạn đặt trong các vòng đơn) để sau này có biện pháp đề phòng các di hại có thể nảy sinh từ tính dư thừa của nó.

Thuộc tính không tính toán (còn được gọi là thuộc tính nguyên thủy) là thuộc tính mà giá trị của nó được cung cấp từ bên ngoài hệ thống (không thể suy ra được từ bên trong hệ thống).

Thuộc tính bền vững là thuộc tính mà giá trị của nó khi đã có thì không thay đổi được nữa. Ngược lại là thuộc tính không bền vững.

Phối hợp hai tiêu chí trên, ta có bốn loại thuộc tính:

- Thuộc tính tính toán và bền vững, chẳng hạn: Tổng tiền của một Hóa đơn, Tiền lương (của một nhân viên trong tháng hiện tại).
- Thuộc tính tính toán và không bền vững, chẳng hạn: Lượng hàng tồn kho, Số dư tiết kiệm.
- Thuộc tính không tính toán và bền vững, chẳng hạn: Ngày sinh, Giới tính (của một Người).
- Thuộc tính không tính toán và không bền vững, chẳng hạn: Số các con của của một nhân viên, Địa chỉ một khách hàng.

đ) Các thuộc tính khóa, thuộc tính kết nối và các liên kết

Một kiểu thuộc tính khóa của một kiểu thực thể là một kiểu thuộc tính mà giá trị của nó tương ứng với mỗi thực thể là riêng biệt cho thực thể đó. Vậy khóa cho phép chỉ định mỗi thực thể, cho phép phân biệt các thực thể với nhau.

Trong bước đầu phân tích, có thể chưa cần quan tâm vội đến việc chọn khóa. Song rốt cục thì trong lược đồ dữ liệu hoàn chỉnh, mỗi kiểu thực thể đều phải có khóa.

Thuộc tính khoá có thể chọn trong số các thuộc tính tự nhiên, miễn là giá trị của nó là khác biệt cho mỗi thực thể. Chẳng hạn nếu tên chính thức cho mỗi công ty đều đã được đăng ký với nhà nước và bảo đảm không trùng lắp, thì có thể lấy Tên công ty làm khóa cho các công ty. Nếu trong một khách sạn, các phòng được sơn màu khác nhau, thì có thể lấy Màu làm khóa chỉ định các phòng trong khách sạn đó.

Tuy nhiên, ít khi tìm được một thuộc tính tự nhiên làm khóa, bấy giờ người ta dùng thuộc tính khóa nhân tạo. Ấy là các mã, các số hiệu, như là Mã hàng, Số hiệu đơn hàng...

Trong một kiểu thực thể, một kiểu thuộc tính được gọi là kiểu thuộc tính kết nối (hay còn gọi là một *tham chiếu*) nếu nó không phải là khóa của kiểu thực thể đó, nhưng lại là khóa của một kiểu thực thể khác. Như ta đã biết, thì tham chiếu là biểu hiện (hay là sự cài đặt) của một kiểu liên kết (1-nhiều) giữa hai kiểu thực thể.

Sau khi đã có tập hợp các kiểu thực thể, thì ta cần nghiên cứu phát hiện các kiểu liên kết giữa các kiểu thực thể trong đó để đưa vào lược đồ.

Việc tìm các kiểu liên kết hai ngôi có thể làm theo 2 cách:

- Dựa trên ý nghĩa của các kiểu thực thể (thực thể trong đó, có thể đóng vai trò gì?), dựa theo các quy tắc quản lý hay các quy trình giao dịch mà ta phát hiện ra các kiểu liên kết.

Thí dụ: Quy trình bán hàng bắt đầu bằng việc một khách hàng đưa đến một đơn đặt hàng. Từ sự kiện này ta phát hiện ra có mối liên quan giữa khách hàng và đơn hàng.

- Tìm trên danh sách các kiểu thuộc tính của các kiểu thực thể (đã lập ở bước trước) hoặc tìm trên các chứng từ giao dịch, ta có thể phát hiện ra các tham chiếu. Mỗi tham chiếu là biểu hiện cho một kiểu liên kết.

Thí dụ: Xem các Đơn hàng, ta thấy Đơn hàng nào cũng có Tên (hay Mã số khách hàng). Đó chính là một tham chiếu. Vậy phải có một kiểu liên kết giữa hai kiểu thực thể Đơn hàng và Khách hàng.

Sau khi phát hiện các kiểu liên kết hai ngôi, ta cần nghiên cứu kỹ các ứng số của chúng, làm căn cứ cho việc vận dụng các Quy tắc 5 và 6 để thực hiện các biến đổi về mô hình E/A hạn chế.

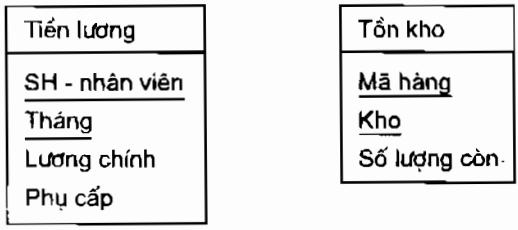
Các kiểu liên kết nhiều ngôi, ít gặp hơn, nhưng ta cũng cần rà soát phát hiện chúng để đưa vào mô hình.

e) Gom nhóm các kiểu thuộc tính thành các kiểu thực thể hay các kiểu liên kết

Cuối cùng thì trong mô hình E/A hạn chế, ta có các kiểu thuộc tính gom thành từng nhóm theo các kiểu thực thể hay các kiểu liên kết (đã thực thể hóa). Ta có các trường hợp như sau:

- Nhóm gồm một thuộc tính khóa và ít nhất một thuộc tính mô tả (có thể có thêm tham chiếu): đó là một kiểu thực thể.
- Nhóm gồm một khóa bội, tạo thành từ khóa của nhiều kiểu thực thể khác, ngoài ra có thể có các thuộc tính mô tả: đó là một kiểu liên kết (đã thực thể hóa).

Trong khóa bội, đôi khi có thể có mặt một thuộc tính trỏ địa điểm hay thời gian, không phải là khóa của kiểu thực thể nào cả. Thành phần của khóa đó gọi là thuộc tính không - thời gian (Hình IV. 28).



Hình IV.28 Khóa bội có chứa thuộc tính không - thời gian

Một trường hợp đặc biệt của kiểu liên kết (đã thực thể hóa), ấy là kiểu thực thể phụ thuộc. Khóa của nó luôn luôn là khóa bội và bao gồm một thành phần là khóa của kiểu thực thể chính. Một thực thể phụ thuộc không thể tồn tại khi thực thể chính của nó bị loại bỏ.

- Nhóm chỉ gồm một kiểu thuộc tính (cũng là khóa), không có thuộc tính mô tả, không có tham chiếu. Đó là một tham số.

Thí dụ: - Ngày hôm nay

- Mức lương tối thiểu.

Tới đây, ta có thể áp dụng những điều nói ở trên vào Hệ Cung ứng vật tư (thí dụ xuyên suốt) để lập lược đồ khái niệm về dữ liệu cho nó.

Thí dụ QL: Hệ Cung ứng vật tư (tiếp theo).

(1) Trước hết lập lược đồ dữ liệu theo mô hình E/A mở rộng, xem như một phác thảo, chưa cần hoàn chỉnh (Hình IV.29).

Các kiểu thực thể có thể sơ bộ phát hiện là:

- Về nhân lực, tài nguyên có Phân xưởng, Người cung cấp, Mặt hàng;

- Về giao dịch có Dự trù, Đơn hàng, Giao hàng, Hóa đơn, Phát hàng;

- Về các thông tin cấu trúc hóa (sổ sách ghi chép) có Dự trù/ Đơn hàng, Tồn kho, Xuất nhập kho, Người cung cấp/Mặt hàng.

Tiếp đó ta phát hiện các kiểu liên kết giữa chúng bằng cách duyệt lại quá trình giao dịch (từ khi Phân xưởng đưa Dự trù đến khi Phân xưởng nhận được hàng) qua đó ta sẽ thấy được các mối liên hệ giữa các kiểu thực thể. Chú ý rằng Dự trù/Đơn hàng và Người cung cấp/Mặt hàng thực chất là các kiểu liên kết.

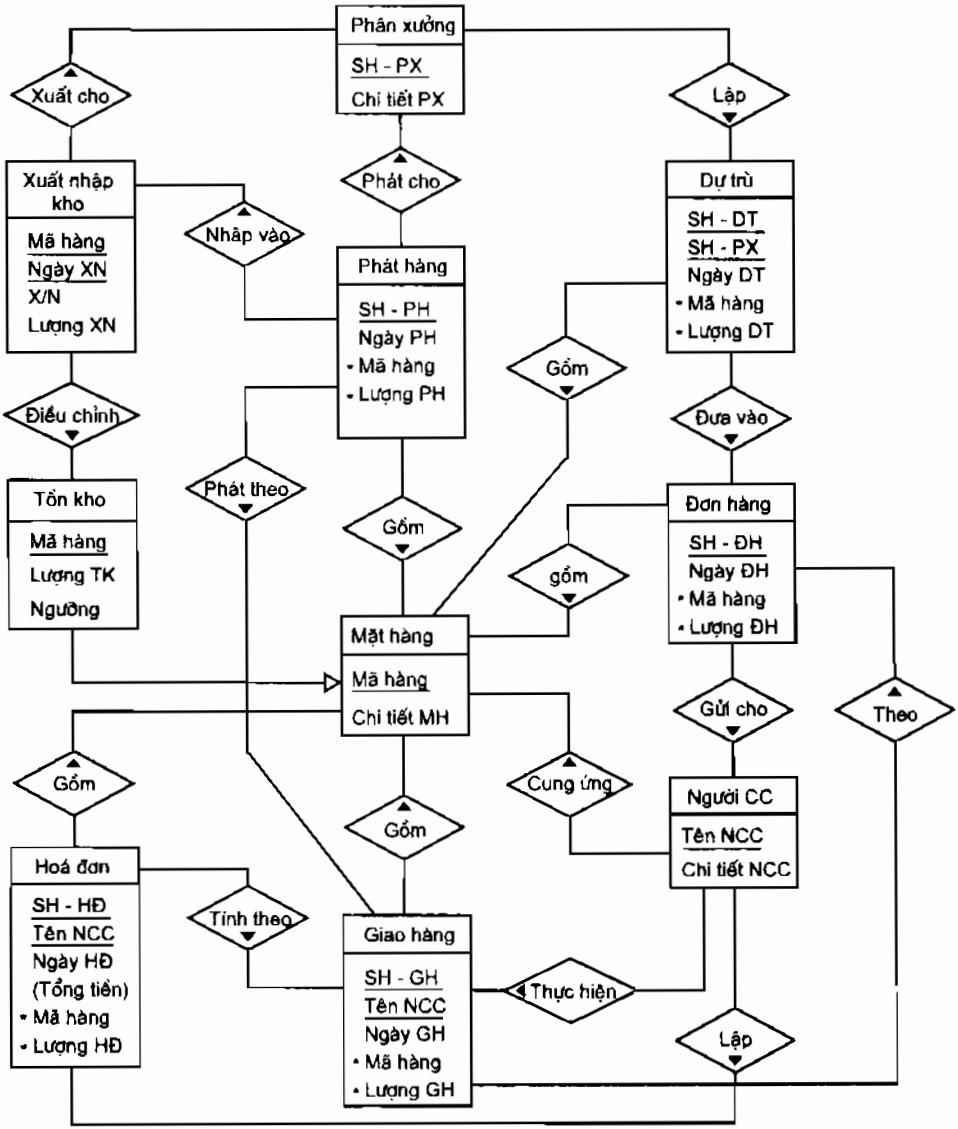
Bổ sung các kiểu thuộc tính cho mối kiểu thực thể (trên hình không ghi chi tiết các thuộc tính mô tả). Riêng việc chọn khóa, ta lưu ý một số điểm sau: Các SH-đơn hàng, SH-giao hàng, SH-phân xưởng, Mã hàng đã được quản lý thống nhất, vậy có thể lấy làm khóa (đơn) của các kiểu thực thể tương ứng. Trái lại SH- dự trù, SH- giao hàng, SH-hóa đơn là ngoại lai đối với hệ thống, có thể xảy ra trùng lắp, vậy muốn dùng chúng làm khóa thì SH- dự trù phải kèm với SH-phân xưởng, SH- giao hàng và SH - hóa đơn phải kèm Tên người cung cấp. Ta xem Tên người cung cấp là không có trùng lắp, nên có thể lấy làm khóa cho kiểu thực thể Người cung cấp được.

Trong Hình IV.29, ta chưa xác định các ứng số vội, vì các kiểu liên kết còn thay đổi ở bước sau; ta dùng một tam giác nhỏ ▸ để chỉ hướng đọc tên các kiểu liên kết và ta dùng một dấu * để đánh dấu các thuộc tính đa trị.

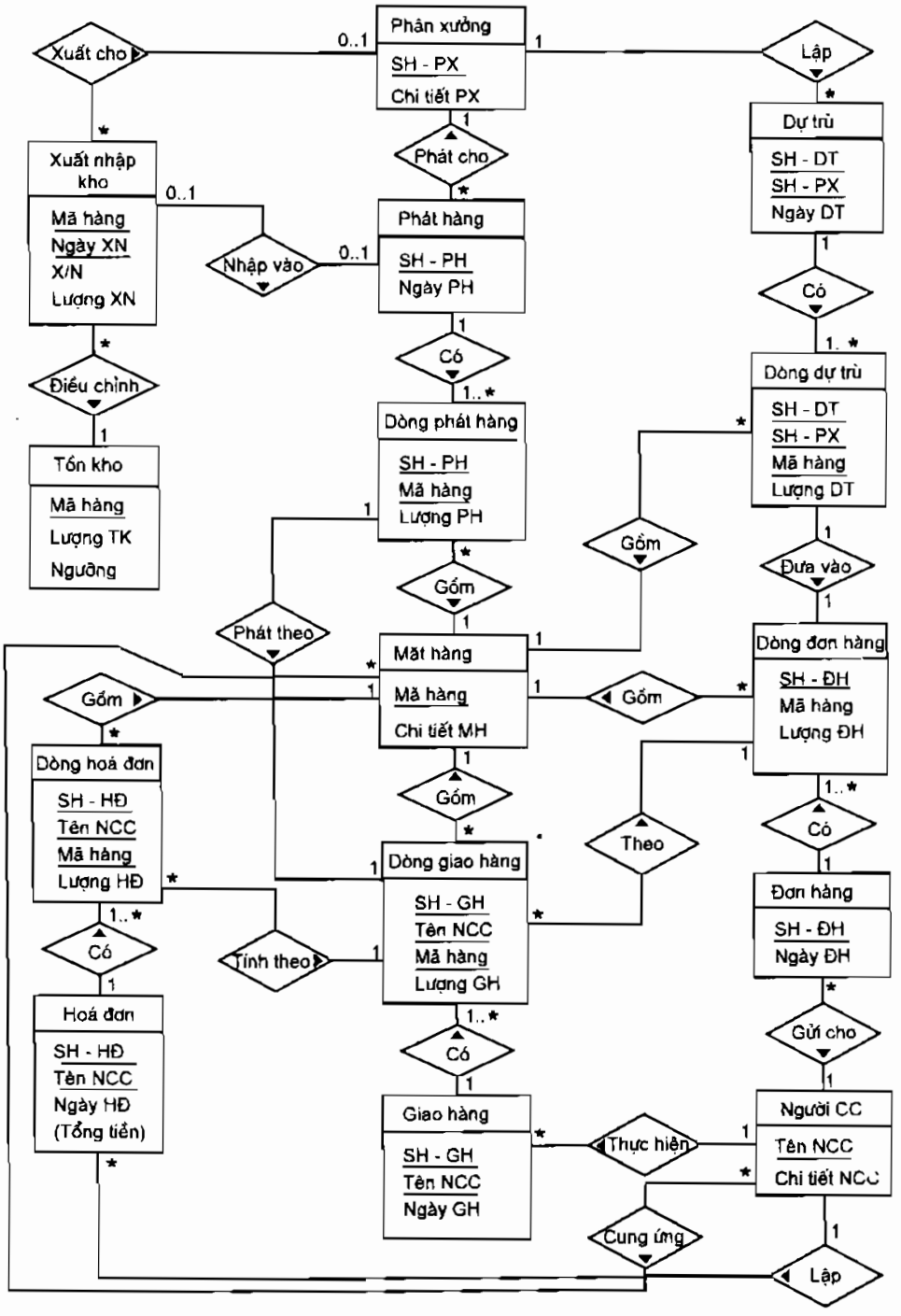
(2) Tiếp đến ta biến đổi từ mô hình E/A mở rộng sang mô hình E/A kinh điển (Hình IV.30). Các kiểu thực thể phụ thuộc được tách ra thay cho các kiểu thuộc tính đa trị. Các kiểu liên kết đã được chỉnh sửa lại. Các ứng số được xác định. Mối liên hệ khái quát hóa giữa Tồn kho và Mặt hàng được bỏ qua (cho đỡ rườm) vì ý nghĩa của nó đã được ẩn chứa trong khóa chung của chúng là Mã hàng rồi.

(3) Cuối cùng mô hình E/A kinh điển được chuyển sang mô hình E/A hạn chế. Mỗi kiểu liên kết 1-1 hay 1- nhiều đều được thể hiện lại bằng đường nối có chân vịt (bỏ tên, bỏ ứng số), với một kiểu thuộc tính kết nối được bổ sung (nếu chưa có) vào đầu "nhiều". Kiểu liên kết nhiều - nhiều duy nhất là Cung ứng được thực thể hóa thành Người CC/Mặt hàng (thêm kiểu thuộc tính mô tả Đơn

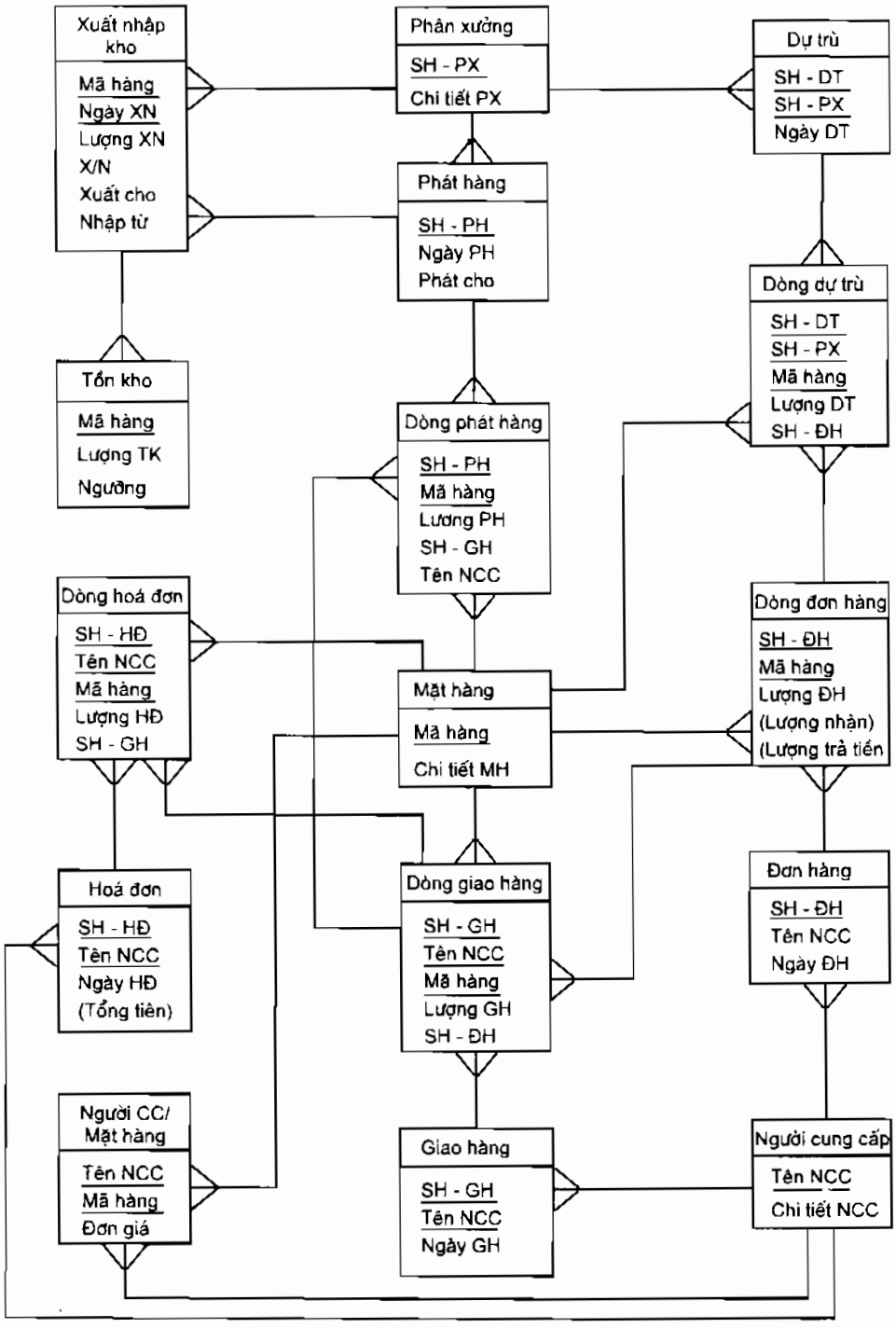
giá). Dòng Đơn hàng được bổ sung hai kiểu thực thể tình thế (đặt trong vòng đơn) là Lượng nhận và Lượng trả tiền để dễ theo dõi quá trình thực hiện đơn hàng (nhớ lại rằng người ta cho phép giao hàng nhiều lần, trả tiền nhiều lần cho một khoản đặt hàng). Kiểu thực thể Dự trù/ Đơn hàng vốn có vai trò quan trọng trong việc tìm địa chỉ phát hàng (xem biểu đồ luồng dữ liệu ở Hình III.34) thì nay chính là kiểu thực thể Dòng dự trù, sau khi bổ sung thêm một kiểu thuộc tính kết nối là SH - ĐH.



Hình IV.29 Lược đồ dữ liệu theo mô hình E/A mở rộng



Hình IV.30 Lược đồ dữ liệu theo mô hình E/A kinh điển



Hình IV.31 Lược đồ dữ liệu theo mô hình E/A hạn chế

Chương V
PHÂN TÍCH HỆ THỐNG VỀ DỮ LIỆU
(bước hoàn chỉnh)

Mô hình E/A trình bày ở chương trước là dễ lập vì nó phản ánh sát sao các đối tượng trong thực tế. Tuy nhiên mô hình E/A (kể cả mô hình E/A hạn chế) chưa phải là đã tránh được các nhược điểm (đặc biệt là tính dư thừa). Cho nên qua bước sơ bộ dùng mô hình E/A để lập lược đồ dữ liệu, ta còn phải hoàn chỉnh tiếp lược đồ đó bằng cách:

- chuyển qua mô hình quan hệ để thực hiện sự chuẩn hóa;
- bổ sung các ràng buộc toàn vẹn, tức là các điều kiện mà lược đồ dữ liệu phải thỏa mãn.

Chương này sẽ trình bày bước hoàn chỉnh đó với hai phần: mô hình quan hệ và các ràng buộc toàn vẹn.

§1. MÔ HÌNH QUAN HỆ

Mô hình quan hệ do Codd đề xuất năm 1970, với các ưu điểm như sau:

- Đơn giản: các dữ liệu được biểu diễn dưới một dạng duy nhất, là quan hệ, tức là các bảng giá trị, khá tự nhiên và dễ hiểu đối với người dùng không chuyên tin học;
- Chặt chẽ: các khái niệm được hình thức hóa cao, cho phép áp dụng các công cụ toán học, các thuật toán;
- Trừu tượng hóa cao: mô hình chỉ dừng ở mức quan niệm, nghĩa là độc lập với mức vật lý, với sự cài đặt, với các thiết bị. Nhờ đó làm cho tính độc lập giữa dữ liệu và chương trình cao.
- Cung cấp các ngôn ngữ truy nhập dữ liệu ở mức cao (như SQL,...), dễ sử dụng và trở thành chuẩn.

Ở đây, ta sử dụng mô hình quan hệ như là một bước tiếp nối để hoàn chỉnh các lược đồ dữ liệu đã lập theo mô hình E/A. Nội dung trình bày chỉ vừa đủ

cho mục đích đó. Bạn đọc muốn tìm hiểu kỹ hơn về mô hình quan hệ, xin xem các sách về các hệ QT CSDL quan hệ của các tác giả: Date, Ullman, Adiba, Gardarin,...

1. Các định nghĩa cơ bản

a) Quan hệ

Cho $D_1, D_2, ..., D_n$ là n miền các giá trị, không nhất thiết khác nhau. Theo định nghĩa toán học thì:

- *tích Đề các* $D_1 \times D_2 \times .... \times D_n$ của n miền trên là tập tất cả các bộ-n có dạng $<d_1, d_2, ..., d_n>$ có thể lập được sao cho $d_i \in D_i$ (i= 1.. n).

- một *quan hệ* R trên các miền $D_1, D_2, ..., D_n$ là một tập con của tích Đề các giữa các miền đó, nghĩa là:

$$R \subseteq D_1 \times D_2 \times ... \times D_n$$

Nếu viết ra tất cả các bộ-n của quan hệ R, mỗi bộ trên một dòng, các thành phần cùng vị trí trên các bộ được gióng thẳng đứng với nhau, thì ta có một bảng hai chiều: số n các cột được gọi là *cấp* của quan hệ, số m các dòng được gọi là *bản số* của quan hệ.

Thí dụ: Cho D_1 là tập các người và D_2 là tập các số nguyên dương. Quan hệ $R \subseteq D_1 \times D_1 \times D_2$, trong đó mỗi bộ <a, b, k > diễn tả một cuộc hôn nhân (anh a lấy chị b vào năm k) có thể biểu diễn thành bảng như sau:

Cột thứ nhất	Cột thứ hai	Cột thứ 3
Giáp	Lan	1972
Ất	Tuyết	1980
Bính	Hồng	1985

b) Các thuộc tính

Theo định nghĩa toán học của quan hệ, thì các miền $D_1, D_2, ..., D_n$ là có thứ tự. Chúng không thể trao đổi vị trí cho nhau, vì vai trò của chúng trong sự diễn tả ý nghĩa của quan hệ là gắn với vị trí. Chẳng hạn ở thí dụ trên thì cột thứ nhất được diễn tả là chồng, cột thứ hai được diễn tả là vợ và cột thứ ba được diễn tả là năm kết hôn.

Tráo đổi vị trí thì ý nghĩa của quan hệ sẽ sai lạc đi. Tuy nhiên nếu ta đem tên vai trò của mỗi cột gắn cho cột đó (xem là tên của cột), thì vị trí lại không còn là quan trọng nữa. Gọi tên đó của mỗi cột là một *thuộc tính*. Trong tin học, ta xem ý nghĩa của mỗi cột là được giải thích bởi tên của cột đó (thuộc tính), chứ không phải bởi vị trí của cột đó, như trong định nghĩa toán học.

Thí dụ: Quan hệ hôn nhân ở trên có thể được viết lại với các thuộc tính thay cho số thứ tự cột như sau:

Chồng	Vợ	Năm kết hôn
Giáp	Lan	1972
Ất	Tuyết	1980
Bính	Hồng	1985

c) Lược đồ quan hệ

Như trên đã thấy, thì một quan hệ là một bảng hai chiều các giá trị, đó là tập hợp các bộ của quan hệ tại một thời điểm nào đó. Tuy nhiên quan hệ lại thường chịu các phép cập nhật (bổ sung, loại bỏ, điều chỉnh các bộ), do đó quan hệ thay đổi theo thời gian.

Tuy thay đổi về giá trị qua cập nhật, song quan hệ vẫn phải giữ một số đặc điểm về ngữ nghĩa. Ta nói là nó vẫn tuân thủ một lược đồ quan hệ nhất định.

Ta định nghĩa, một *lược đồ quan hệ* là sự hợp thành của hai yếu tố:

- một cấu trúc, gồm tên quan hệ và một danh sách các thuộc tính (mỗi thuộc tính gắn với một miền), thường cho dưới dạng: $R (A_1, A_2, ..., A_n)$.
- một tập hợp các ràng buộc toàn vẹn, tức là các điều kiện mà mọi quan hệ trong lược đồ đều phải thỏa mãn.

Như vậy, có thể nói lược đồ quan hệ là một khuôn mẫu, hay nói đúng hơn là một cái lọc, cho phép thiết lập một họ các quan hệ cùng ngữ nghĩa.

Thí dụ: Một lược đồ quan hệ:

THỜI-GIAN-BIỂU (LỚP-SV, NGÀY, GIỜ, PHÒNG, THẦY) với các ràng buộc toàn vẹn được cho chẳng hạn như sau:

(a) Một bộ <l, n, g, p, t> có nghĩa là: "lớp sinh viên *l*, vào ngày n, giờ g, tại phòng học p học với thầy t".

(b) Miền của thuộc tính GIỜ là tập hợp các số nguyên từ 7 đến 21.

(c) Vào một ngày và giờ đã cho thì một thầy giáo không thể đồng thời ở trong hai phòng.

Ghi chú: Dù định nghĩa như trên, song sau này, trong một khung cảnh không thể có nhầm lẫn, ta thường gọi tắt theo thói quen như sau:

- Gọi R (A1, A2, ..., An) là lược đồ quan hệ (tạm gác các ràng buộc toàn vẹn sang bên).

- Thậm chí dùng thuật ngữ "quan hệ" thay vì lược đồ quan hệ.

d) Đối sánh quan hệ với kiểu thực thể hay kiểu liên kết

Ta biết trong mô hình E/A, thì mỗi kiểu thực thể (hay kiểu liên kết đã thực thể hóa) là một tập hợp các thực thể được mô tả bởi cùng một tập hợp các kiểu thuộc tính. Vậy nếu liệt kê tất cả các thực thể đó với các giá trị thuộc tính của chúng, ta có một bảng 2 chiều, trong đó mỗi cột tương ứng một kiểu thuộc tính, mỗi dòng diễn tả một thực thể. Gọi đó là *biểu diễn bảng* của kiểu thực thể (hay kiểu liên kết), và ta thấy đó chính là một quan hệ.

Tuy nhiên biểu diễn bảng của kiểu thực thể ít khi được làm rõ, vì trong mô hình hóa ít khi ta phải quan tâm cụ thể tới các giá trị thuộc tính. Bởi vậy thay vì một bảng, ta thường biểu diễn kiểu thực thể (hay kiểu liên kết đã thực thể hóa) dưới dạng một hình chữ nhật chứa tên và danh sách các kiểu thuộc tính. So sánh với mô hình quan hệ, thì đó chính là một lược đồ quan hệ (xem Hình V.1).

Khách hàng
SH - KH
Tên KH
Địa chỉ KH
Tài khoản KH

SH - KH	Tên KH	Địa chỉ KH	Tài khoản KH
315	Hoàng	30 Đại La	4798
316	Bích	40 Ngõ Gạch	4729
318	Lan	15 Kim Ngưu	3792
330	Tuyết	40 Hàng Bài	1738

Khách hàng (SH-KH, Tên-KH, Địa chỉ-KH, Tài khoản-KH)

Hình V.1 Kiểu thực thể, biểu diễn bảng của nó (dưới dạng quan hệ) và lược đồ quan hệ tương ứng.

Nhưng ngược lại, thì phải chăng mọi quan hệ đề cập trong mô hình quan hệ đều là kiểu thực thể (hay kiểu liên kết đã thực thể hóa)? Không phải như vậy.

Quan hệ có thể có ý nghĩa rộng lớn hơn, có khi là phức tạp và thậm chí rối rắm hơn. Thí dụ Hình V.2 cho một quan hệ là một Bảng chấm công, mà mỗi bộ <c, m, t, p, x> trong đó được hiểu là: "Công nhân c làm việc trên máy m một thời gian t, trong phân xưởng p do ông x làm trưởng phân xưởng". Thoạt nhìn thì có thể nói quan hệ này diễn tả kiểu liên kết "Làm việc" của công nhân trên máy (xác định bởi 2 cột đầu). Song nếu Thời gian quả là thuộc tính của sự làm việc đó, thì SH-PX khó nói là thuộc tính của sự làm việc đó, vì thực ra nó là thuộc tính của Máy (địa điểm đặt máy); còn Trưởng PX thì lại còn xa lạ hơn với sự làm việc đó, vì thực ra nó là thuộc tính của Phân xưởng (người phụ trách của phân xưởng). Cho nên bảng này không phải là một kiểu liên kết đúng theo định nghĩa của mô hình E/A hạn chế.

CHẤM CÔNG:

Mã - CN	SH - Máy	Thời gian	SH - PX	Trưởng - PX
C1	m1	10	p1	Giáp
C1	m2	10	p1	Giáp
C2	m3	50	p2	Ất
C3	m5	100	p3	Đinh
C3	m4	30	p2	Ất
C2	m2	20	p1	Giáp

Hình V.2 Bảng chấm công dưới dạng một quan hệ

2. Chiếu, kết nối và phân rã các quan hệ

Có nhiều phép toán có thể tiến hành trên các quan hệ để cho các quan hệ. Ở đây ta chỉ xét một số phép toán có liên quan tới việc chuẩn hóa quan hệ mà ta sẽ xét ở phía dưới.

a) Phép chiếu

Mục đích của phép này là nhằm loại bỏ một số thuộc tính và thu hẹp quan hệ về các thuộc tính còn lại.

Giả sử U là tập các thuộc tính của quan hệ R và $V \subseteq U$. *Chiếu* của R trên V là tập mọi bộ trên V sao cho tồn tại một bộ của R trùng với nó trên các thuộc tính của V.

Ký hiệu: $\pi_V(R)$

Thí dụ: Cho R (U) với U = (SH - KH, Loại - KH, Miễn giảm)
và cho V = (Loại - KH, Miễn giảm)

Với R(U):

SH - KH	Loại - KH	Miễn giảm
1	2	1,5
2	1	0,5
3	2	1,5
4	3	2

thì có: $\Pi_V(R)$:

Loại - KH	Miễn giảm
2	1,5
1	0,5
3	2

Chú ý: Sau khi xóa các cột thừa, phải loại bỏ bớt các dòng đồng nhất.

b) Kết nối hai quan hệ

Mục đích của phép kết nối (còn gọi là kết nối tự nhiên) là ghép nối mọi bộ của quan hệ R với mọi bộ của quan hệ S, nếu hai bộ đó có cùng giá trị trên các thuộc tính chung của R và S, có loại bỏ sự trùng lắp đối với các thuộc tính chung. Ký hiệu: $R \bowtie S$

Thí dụ: Cho hai quan hệ:

R:

SH - ĐH	Ngày - ĐH	SH - KH
3	10-1-99	2
4	10-1-99	5
5	10-1-99	3

S:

SH - KH	Tên
1	Tý
2	Sửu
3	Dần
4	Mão

thì kết quả kết nối hai quan hệ đó là:

$R \bowtie S$

SH - ĐH	Ngày - ĐH	SH - KH	Tên
3	10-1-99	2	Sửu
5	11-1-99	3	Dần

c) Phân rã một quan hệ

Mục đích là tách một quan hệ R thành hai quan hệ S và T nhỏ hơn mà không mất mát thông tin, tức là:

i) S và T đều là chiếu của R,

ii) Kết nối của S và T lại là R.

Thí dụ:

R:

Lớp	Môn	Sinh viên	Thầy
d1	Đại số	Hùng	Long
d1	Đại số	Nam	Long
g1	Giải tích	Hoa	Bình
g1	Giải tích	Lan	Bình
d2	Đại số	Tiến	Long
d2	Đại số	Đạt	Long

Có thể phân rã R thành hai quan hệ:

R1:

Lớp	Sinh viên
d1	Hùng
d1	Nam
g1	Hoa
g1	Lan
d2	Tiến
d2	Đạt

R2:

Lớp	Môn	Thầy
d1	Đại số	Long
g1	Giải tích	Bình
d2	Đại số	Long

Cũng có thể có cách phân rã khác, chẳng hạn:

R3:

Lớp	Môn	Sinh viên
d1	Đại số	Hùng
d1	Đại số	Nam
g1	Giải tích	Hoa
g1	Giải tích	Lan
d2	Đại số	Tiến
d2	Đại số	Đạt

R4:

Lớp	Thầy
d1	Long
g1	Bình
d2	Long

Ngược lại có những quan hệ là không phân rã được.

Chẳng hạn:

R:

SH - HĐ	Mã hàng	Số lượng	Thành tiền
1	10	5	50
1	20	8	120
2	10	15	150
2	30	5	50

Chiếu R lần lượt lên các cụm thuộc tính {SH - HĐ, Số lượng, Thành tiền} và {Mã hàng, Số lượng}, ta thu được hai quan hệ:

S:

SH - ĐH	Số lượng	Thành tiền
1	5	50
1	8	120
2	15	150
2	5	50

T:

Mã hàng	Số lượng
10	5
20	8
10	15
30	5

Lại đem kết nối S với T, ta sẽ được một quan hệ xác định trên cùng một tập các thuộc tính với R, nhưng khác R vì nó có chứa các dòng mới, thí dụ kết

nối dòng đầu của S với dòng cuối của T (cùng Số lượng là 5) ta có dòng <1, 30, 5, 50> vốn không có trong R.

Thử trên các cụm thuộc tính khác của R, ta vẫn không thành công, vậy quan hệ R là không phân rã được.

3. Phụ thuộc hàm

Dưới đây ta dùng một số quy ước trong cách viết như sau:

- dùng các chữ lớn A, B, C,... để trỏ các tập thuộc tính; A ∪ B sẽ viết là A,B.
- dùng các chữ nhỏ (có thể có chỉ số) để trỏ giá trị của các thuộc tính, chẳng hạn a_1 là giá trị của A (gồm các giá trị của các thuộc tính trong A).

- dùng móc nhọn để chỉ các bộ, chẳng hạn $<a_1, b_1, c_1>$ là một bộ của quan hệ R (A,B,C).

a) Định nghĩa phụ thuộc hàm

Cho quan hệ R (A,B,C), trong đó C có thể rỗng. Ta nói tập các thuộc tính B là *phụ thuộc hàm* vào tập các thuộc tính A, nếu trong R bất cứ hai bộ $<a_1, b_1, c_1>$, $<a_2, b_2, c_2>$ nào mà có $a_1 = a_2$, thì cũng đều có $b_1 = b_2$. Nói cách khác, luôn luôn có cùng một giá trị của B đi liền với một giá trị cho trước của A trong quan hệ R (cho nên cũng nói là: *A xác định B*).

Ký hiệu phụ thuộc hàm: $A \rightarrow B$

Thí dụ: Trở lại quan hệ Chấm công cho ở Hình V.2, ta thấy có một số các phụ thuộc hàm như sau:

Mã-CN, SH - Máy → Thời gian, SH - PX, Trưởng - PX

SH - Máy → SH - PX, Trưởng - PX

SH - PX → Trưởng - PX

Chú thích:

+ Phụ thuộc hàm được định nghĩa dựa trên hiện tượng trùng hợp về giá trị của các bộ trong một quan hệ (bảng giá trị) đã cho sẵn. Tuy nhiên sự trùng hợp giá trị đó nói chung không phải là ngẫu nhiên, mà là phản ánh một mối liên quan trong thực tế. Ví dụ SH-PX → Trưởng-PX phản ánh một thực tế là: mỗi phân xưởng có một trưởng phân xưởng và chỉ một mà thôi.

+ Phụ thuộc hàm đã được định nghĩa cho quan hệ (bảng giá trị), chứ không phải cho lược đồ quan hệ. Tuy vậy sau này ta vẫn nói phụ thuộc hàm cho một lược đồ quan hệ, hiểu theo nghĩa là: phụ thuộc hàm này phải đúng cho mọi quan hệ trong lược đồ quan hệ đó. Bấy giờ phụ thuộc hàm thực chất là một ràng buộc toàn vẹn (điều kiện phải thỏa mãn) của lược đồ quan hệ.

b) Phụ thuộc hàm sơ đẳng, phụ thuộc hàm trực tiếp và khóa của quan hệ

- Một phụ thuộc hàm A→B là phụ thuộc hàm *sơ đẳng* nếu không tồn tại A'⊂ A mà A'→B. Nói cách khác không có thuộc tính thừa trong vế trái của phụ thuộc hàm.

Chẳng hạn trong quan hệ Chấm công nói trên thì:

Mã - CN, SH - máy → Thời gian là PTH sơ đẳng

còn Mã - CN, SH - máy → SH - PX không phải là PTH sơ đẳng,

vì có: SH - máy → SHPX

- Một phụ thuộc hàm A→B trong một quan hệ R là phụ thuộc hàm *trực tiếp* nếu không tồn tại tập thuộc tính C trong R khác với A và B, mà: A→C và C→B.

Chẳng hạn trong quan hệ Chấm công nói trên thì:

SH - PX → Trưởng - PX là PTH trực tiếp

SH - máy → Trưởng - PX không phải là PTH trực tiếp

vì: SH - máy → SH - PX

và SH - PX → Trưởng - PX

- A là *khóa* của một quan hệ R (A,B) nếu A→B là một phụ thuộc hàm sơ đẳng trong R. Nói cách khác:

+ Giá trị của A xác định duy nhất giá trị của các thuộc tính còn lại, tức là xác định duy nhất một bộ, và

+ Trong A không có thuộc tính thừa (A là tối tiểu).

c) Tính chất của các phụ thuộc hàm

Cho A,B,C,D là các tập thuộc tính. Từ định nghĩa của phụ thuộc hàm, ta dễ dàng kiểm nghiệm các tính chất sau đây của các phụ thuộc hàm. Chúng đều có dạng là các quy tắc suy diễn và được gọi là các tiên đề Armstrong (1974):

(i) Tính phản xạ (reflexivity) : Nếu $B \subseteq A$ thì $A \rightarrow B$

(ii) Tính tăng cường (augmentation) : Nếu $A \rightarrow B$ thì $A, C \rightarrow B$ với C bất kỳ

(iii) Tính bắc cầu (transitivity): Nếu $A \rightarrow B$ và $B \rightarrow C$ thì $A \rightarrow C$

Một phụ thuộc hàm được suy ra từ (i) được gọi là phụ thuộc hàm *tầm thường*, từ (ii) là thuộc hàm *không sơ đẳng*, và từ (iii) là phụ thuộc hàm *không trực tiếp*.

Từ ba tiên đề trên, ta có thể suy ra thêm nhiều tính chất khác. Sau đây là ba tính chất dẫn xuất, song cũng rất tiện dùng và sẽ được nhắc tới luôn như là các tiên đề:

(iv) Tính phân rã (decomposition): Nếu $A \rightarrow B$ và $D \subseteq B$ thì $A \rightarrow D$

(v) Tính gộp (union): Nếu $A \rightarrow B$ và $A \rightarrow C$ thì $A \rightarrow B, C$

(vi) Tính giả bắc cầu (pseudotransitivity) : Nếu $A \rightarrow B$ và $B, D \rightarrow C$ thì $A, D \rightarrow C$

4. Tinh giản một tập hợp các phụ thuộc hàm

Vì các tiên đề Armstrong đều là các quy tắc suy diễn, vậy cho trước một tập F các phụ thuộc hàm, vận dụng các tiên đề đó ta sẽ thu thêm được nhiều phụ thuộc hàm khác.

Nói cách khác, với các tiên đề Armstrong, ta có thể làm lộ diện các phụ thuộc hàm vốn tiềm ẩn trong F. Ngược lại, ta thường cố gắng tìm cách tinh giản tập F sao cho không đánh mất các phụ thuộc hàm tiềm ẩn trong đó. Có nhiều cách để tinh giản F, hoặc làm bằng tay (dựa vào trực quan), hoặc làm bằng máy tính (dựa vào giải thuật) mà mục đích đều là: loại bỏ các phụ thuộc hàm tầm thường, phụ thuộc hàm không sơ đẳng và phụ thuộc hàm không trực tiếp. Ta sẽ xem xét các cách làm này ở phía dưới, sau khi đưa ra vài khái niệm cần thiết.

a) Đồ thị phụ thuộc hàm, bao đóng và phủ tối tiểu

Cho một tập hợp F các phụ thuộc hàm phản ánh các mối liên quan về ngữ nghĩa trên một tập các thuộc tính. Giả thiết mọi phụ thuộc hàm trong F đều có vế phải đơn (gồm một thuộc tính). Nếu không, có thể dùng quy tắc (iv) (tính phân rã) để đưa F về dạng này.

Tập F có thể được biểu diễn dưới dạng một đồ thị có hướng, gọi là *đồ thị phụ thuộc hàm*, như sau:

- mỗi thuộc tính có mặt trong F được biểu diễn thành một nút;
- mỗi vế trái gồm nhiều thuộc tính của một phụ thuộc hàm trong F cũng được biểu diễn thành một nút (thường được vẽ ở phía trên);
- mỗi phụ thuộc hàm trong F được biểu diễn thành một cung vẽ từ nút b ểu diễn cho vế trái tới nút biểu diễn cho vế phải.

Thí dụ: Với các thuộc tính xuất hiện trong quan hệ Chấm công cho ở trong Hình V.2, ta có một tập các phụ thuộc hàm như sau:

Mã - CN, SH - máy → Thời gian

Mã - CN, SH - máy → SH-PX

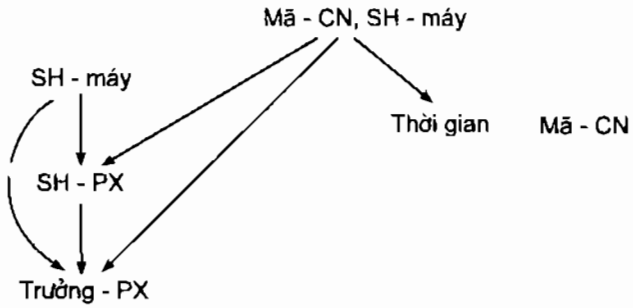
Mã - CN, SH - máy → Trưởng - PX

SH - máy → SH-PX

SH - máy → Trưởng-PX

SH - PX → Trưởng-PX

Đồ thị phụ thuộc hàm biểu diễn tập các phụ thuộc hàm này là như trong Hình V.3.



Hình V.3 Một đồ thị phụ thuộc hàm

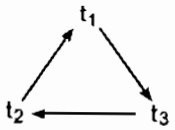
Ta gọi *bao đóng bắc cầu* (gọi tắt là bao đóng) của một tập các phụ thuộc hàm F, ký hiệu là F^+, là tập hợp tất cả các phụ thuộc hàm có thể thu được từ F bằng quy tắc bắc cầu (iii).

Ta gọi *phủ tối tiểu* của một tập hợp F các phụ thuộc hàm, ký hiệu là $\hat{F}$ là một tập hợp các phụ thuộc hàm sơ đẳng có cùng bao đóng với F và điều này không đúng nữa khi ta loại bớt một phụ thuộc hàm bất kỳ trong $\hat{F}$.

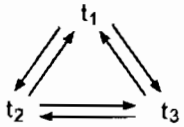
Như vậy một phủ tối tiểu $\hat{F}$ là không rút gọn hơn được nữa sao cho từ đó với các phép (i) - (iii) ta lại có thể khôi phục F và cả F^+. Mục đích của việc tinh giản một tập hợp F các phụ thuộc hàm cho trước, chính là tìm phủ tối tiểu $\hat{F}$ của nó.

Người ta chứng minh rằng nếu đồ thị của F không có chu trình thì mọi phụ thuộc hàm trong $\hat{F}$ đều là phụ thuộc hàm trực tiếp, và ngược lại mọi phụ thuộc hàm trực tiếp trong F đều thuộc $\hat{F}$.

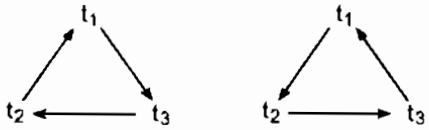
Một thí dụ minh họa trường hợp đồ thị có chu trình là:



Đồ thị này có bao đóng là:



và có hai phủ tối tiểu khác nhau là:



Ta thấy, mọi phụ thuộc hàm trực tiếp trong một đồ thị phủ tối tiểu này lại là không trực tiếp trong đồ thị kia (tức là trong F^+).

Hơn nữa, các phụ thuộc hàm trực tiếp trong đồ thị ban đầu lại đều không có mặt trong dạng thứ hai của phủ tối tiểu của nó.

b) Một số biến đổi trực quan

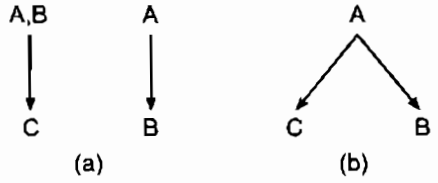
Các biến đổi này đối với một tập F các phụ thuộc hàm cho trước là có tính trực quan (và thiếu chặt chẽ), vì chúng đều dựa trên khả năng quan sát của con người trên đồ thị của F.

• *Loại bỏ phụ thuộc hàm bên trong vế trái của một phụ thuộc hàm*

Giả sử ta có phụ thuộc hàm A,B → C, nhưng mặt khác lại có A → B. Vận dụng các tiên đề Armstrong, ta có:

A → (A, B) → C, và suy ra A → C

Vậy có thể biến đổi đồ thị như trong Hình V.4

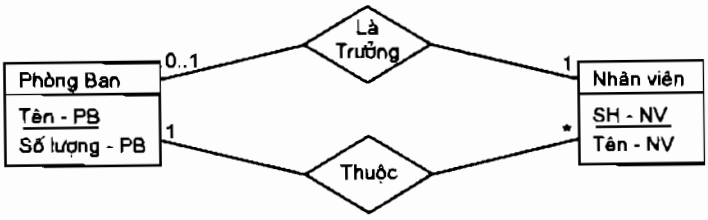


Hình V.4 Loại bỏ phụ thuộc hàm bên trong một nút ((a) thành (b)).

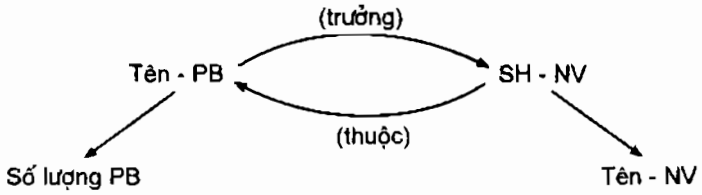
• *Loại bỏ các đường khép kín* (chu trình)

Nếu đồ thị F có chứa đường khép kín, thì nó sẽ gây khó khăn cho quá trình chuẩn hóa sau này. Bởi vậy ta phải loại bỏ sự khép kín đó, bằng cách đưa thêm một thuộc tính mới, đồng nghĩa với một thuộc tính trên chu trình để tách chúng ra.

Chẳng hạn, từ một lược đồ thực thể/ liên kết như trong Hình V.5 ta có một đồ thị phụ thuộc hàm như ở Hình V.6, trong đó có một chu trình.



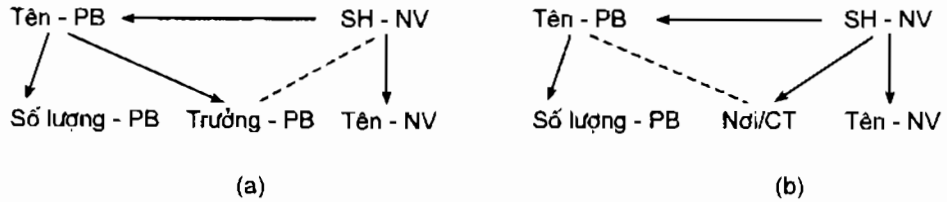
Hình V.5 Hai kiểu liên kết cùng tồn tại giữa hai kiểu thực thể



Hình V.6 Đồ thị phụ thuộc hàm tương ứng

Để cắt đứt chu trình, ta có thể:

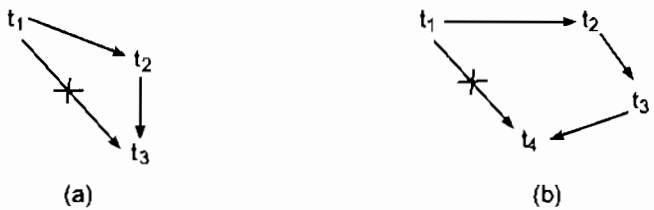
- hoặc là đưa thêm thuộc tính Trưởng-PB đồng nghĩa với SH-NV (tức là cùng miền giá trị), như trong Hình V.7 (a);
- hoặc là đưa thêm thuộc tính Nơi c/t đồng nghĩa với Tên-PB như trong Hình V.7(b).



Hình V.7 Hai cách cắt đứt chu trình

• *Loại bỏ phụ thuộc hàm bắc cầu*

Đó là các phụ thuộc hàm có dạng A → C, mà mặt khác lại có A → B và B → C. Các phụ thuộc hàm bắc cầu là dư thừa, và có thể loại ra khỏi F, vì chúng có thể suy ra từ phần còn lại nhờ tính bắc cầu (iii). Trên đồ thị của F, thì đó là các cung khép kín một hình tam giác, hay tổng quát hơn là khép kín một đường gấp khúc gồm nhiều cung kế tiếp nhau (Hình V.8). Khép kín hiểu theo nghĩa là nối điểm xuất phát và điểm đến của một đường.



Hình V.8 Loại bỏ phụ thuộc hàm bắc cầu

Ghi chú: Phụ thuộc hàm không sơ đẳng có thể xem là một phụ thuộc hàm bắc cầu (và có thể loại bỏ bằng biện pháp trên). Quả vậy A → C là không sơ đẳng nếu tồn tại B ⊂ A mà B → C. Nhưng lúc đó bởi tính chất (i) ta cũng có A → B; vậy A → C là bắc cầu bởi A → B và B → C.

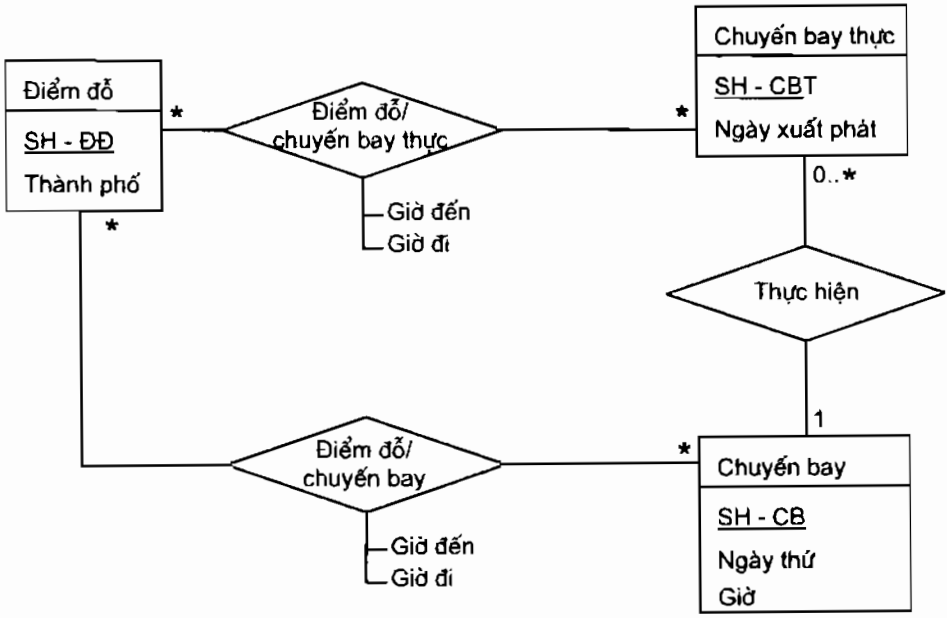
• *Loại bỏ các phụ thuộc hàm giả bắc cầu*

Đó là các phụ thuộc hàm có dạng A, D → C trong khi đã có A → B và B,D → C. Chúng cũng có thể bỏ đi khỏi F vì có thể suy ra được từ phần còn lại bởi tính chất (vi).

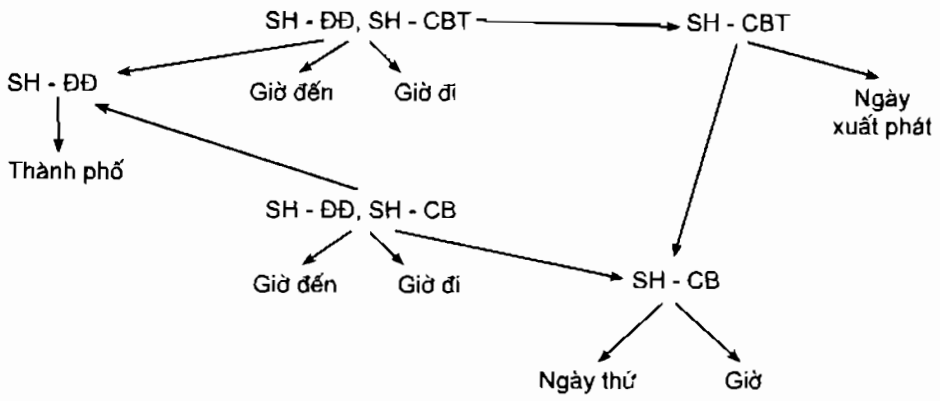
Thí dụ: Trong khi mô hình hóa một hãng hàng không, ta gặp một phần lược đồ E/A như ở Hình V.9, trong đó:

- Chuyến bay: mô tả đặc điểm của các chuyến bay chuẩn thực hiện trong mỗi tuần.
- Chuyến bay thực: mô tả các chuyến bay thực vào các ngày cụ thể.
- Thực hiện: là liên kết nối mỗi chuyến bay thực với một chuyến bay (chuẩn) mà nó thực hiện theo.
- Điểm đỗ: mô tả các nơi mà máy bay đỗ lại trong một chuyến bay.
- Điểm đỗ/ chuyến bay: là danh sách các điểm đỗ cho một chuyến bay (hay ngược lại là danh sách các chuyến bay đáp xuống một điểm đỗ).
- Điểm đỗ/ Chuyến bay thực: là danh sách các điểm đỗ cho một chuyến bay thực (hay ngược lại).

Từ lược đồ E/A nói trên, ta lập được đồ thị các phụ thuộc hàm như trong Hình V.10.



Hình V.9 Lược đồ E/A về các chuyến bay



Hình V.10 Đồ thị các PTH về các chuyến bay

Trên đồ thị ở Hình V.10, ta thấy rằng:

Phụ thuộc hàm: SH - ĐĐ, SH - CBT → Giờ đến, Giờ đi là có thể loại bỏ, vì nó là giả truyền ứng từ hai phụ thuộc hàm:

SH - CBT → SH - CB.

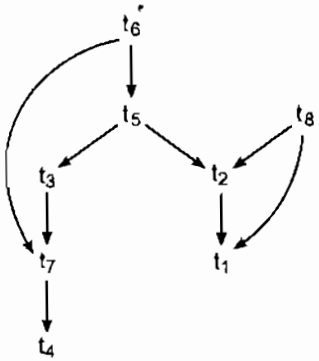
và SH - ĐĐ, SH - CB → Giờ đến, Giờ đi.

c) Các giải thuật tìm bao đóng và phủ tối tiểu

Các biện pháp tinh giản tập hợp các phụ thuộc hàm vừa được trình bày ở trên chỉ có thể thực hiện bởi con người (thủ công) dựa vào khả năng quan sát trên đồ thị. Để có thể sử dụng máy tính, ta phải dựa vào các giải thuật chặt chẽ hơn sẽ nói ở sau đây.

Trong các giải thuật này, ta biểu diễn mỗi đồ thị F dưới dạng một ma trận vuông M, (gọi là ma trận kề) như sau:

- mỗi cột i hay mỗi dòng i tương ứng với một nút t_i, $1 \le i \le n$ (gồm một thuộc tính hay nhóm thuộc tính).
- Phần tử M_{ji} (ở dòng j cột i) là 1 nếu $t_i \rightarrow t_j$ là một phụ thuộc hàm trong F; các phần tử còn lại là 0 (riêng các phần tử trên đường chéo chính đều = 0). Để dễ nhìn, các số 0 sẽ không được viết ra trong thí dụ (Hình V.11).



M :

	1	2	3	4	5	6	7	8
1		1						1
2					1			1
3					1			
4							1	
5						1		
6								
7			1			1		
8								

Hình V.11 Một đồ thị và ma trận kề của nó

• *Giải thuật lập bao đóng*

- Đầu vào: Ma trận M biểu diễn đồ thị các phụ thuộc hàm F

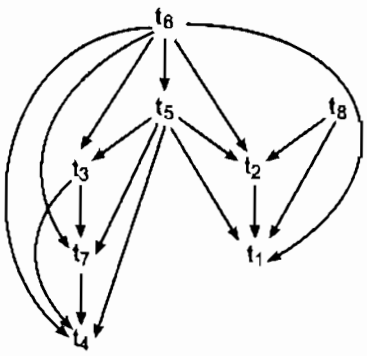
- Đầu ra: Ma trận M' biểu diễn bao đóng bắc cầu F^+ .

- Phương pháp:

{Dùng V là một véctơ làm việc để ghi nhận rằng khi V = C_i thì mọi phần tử trên một cột C_i đang được xử lý là đã được xử lý xong. C_i là cột thứ i, D_j là dòng thứ j của ma trận M. M_{ji} là phần tử nằm ở dòng j, cột i (=1 khi có $t_i \rightarrow t_j$). C'_i là cột thứ i, D'_j là dòng thứ j của ma trận kết quả M', n là cấp của ma trận}

```
BEGIN
   Đọc M ;{ đường chéo chính đưa về 0 }
   WHILE M ≠ 0 DO
      BEGIN
       V ← 0 ;
          Tìm chỉ số i đầu tiên chưa xử lý (modulo n)
          sao cho Li=0 ; { ti không là PTH }
          WHILE V≠Ci DO { xử lý Ci }
              BEGIN
               FOR ALL j : Mji=1 và V(j)=0 DO
                 BEGIN V(j)← 1;
                       FOR ALL k:Mkj=0 DO Mki ← 1
                 END;
                 C'i ← Ci ; { chép Ci đã xử lý }
               END ;
```

```
            Ci ← 0 ; { xóa Ci trong M }
        END;
    Chép ra M'
END.
```



M':

	1	2	3	4	5	6	7	8
1		1			1	1		1
2					1	1		1
3					1	1		
4			1		1	1	1	
5						1		
6								
7			1		1	1		
8								

Hình V.12 Kết quả tìm bao đóng của ma trận M'

• *Giải thuật tìm phủ tối tiểu*

- Đầu vào: Ma trận M' biểu diễn bao đóng F^+ (đường chéo chính đã xóa)

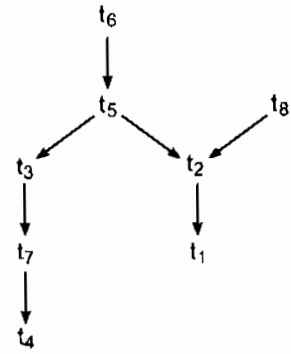
- Đầu ra: Ma trận M'' biểu diễn phủ tối tiểu $\hat{F}$

- Phương pháp:

```
BEGIN
   Đọc M' ;
   WHILE M' ≠ 0 DO
         FOR ALL i: Li = 0 DO
           BEGIN
             FOR ALL k: M'ki = 1  DO
               FOR ALL j: M'kj = M'ji = 1 DO
                     M'ji ←  0; { xóa ti → tj
                     nếu ∃k:ti → tk → tj}
             C''i ← C'i;   { lưu kết quả xử lý}
             C'i ← 0       { xóa C'i trong M' }
           END
   Chép ra M''
END.
```

M" :

	1	2	3	4	5	6	7	8
1		1						
2					1			1
3					1			
4							1	
5						1		
6								
7			1					
8								

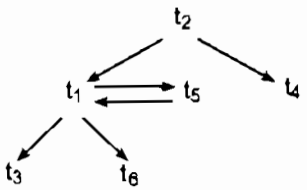


Hình V.13 Kết quả tìm phủ tối tiểu từ ma trận M'

Chú thích:

- Cặp giải thuật trên có lẽ là đơn giản nhất trong 7 hay 8 giải thuật tìm phủ tối tiểu đã có. Tuy nhiên chúng có một hạn chế: đồ thị đầu vào không được có chu trình. Bởi vì nếu có chu trình, sẽ tới một lúc ta không tìm được một dòng trống chưa xử lý và giải thuật bị bế tắc mà thực ra là chưa giải quyết xong (M $\neq$ 0 và chưa ra khỏi vòng lặp).

Chẳng hạn, xuất phát từ đồ thị sau:



thì sau khi xử lý và loại t_2, t_4,... mọi nút còn lại đều là điểm đến của phụ thuộc hàm, vậy không còn dòng trống (chưa xử lý) cho nên giải thuật bị bế tắc, mà chưa xét hết các nút.

- Dễ nhận thấy là 2 giải thuật có hành động ngược nhau (thêm vào và loại bỏ các cung bắc cầu). Vậy phải chăng giải thuật tìm bao đóng là thừa? Không đúng, vì giải thuật 2 nếu áp dụng lên một đồ thị chưa đóng sẽ có thể không loại bỏ hết được các cung bắc cầu. Bạn đọc thử áp dụng giải thuật 2 lên đồ thị cho ban đầu ở Hình V.11 sẽ thấy phụ thuộc hàm $t_6 \rightarrow t_7$ là không loại bỏ được.

- Việc áp dụng liên tiếp hai giải thuật trên chỉ cho phép loại bỏ các phụ thuộc hàm không trực tiếp (bắc cầu). Đối với các phụ thuộc hàm không sơ đẳng ta cũng cần loại bỏ chúng. Đó là các phụ thuộc hàm có dạng $A \rightarrow C$ mà đồng thời lại đã có phụ thuộc hàm $B \rightarrow C$, với $B \subset A$. Vậy muốn cho cặp giải thuật trên loại bỏ được phụ thuộc hàm không sơ đẳng $A \rightarrow C$, thì trên đồ thị phải có mặt cung $A \rightarrow B$ (phụ thuộc hàm tầm thường) để tạo thành một tam giác với $A \rightarrow C$ và $B \rightarrow C$. Vậy ta áp dụng mẹo sau: Trước khi xuất phát, hãy xem xét các nút của đồ thị, nếu $t_j \subset t_i$ thì thêm cung $t_i \rightarrow t_j$ (thêm phụ thuộc hàm tầm thường). Sau khi áp dụng xong giải thuật, loại bỏ các cung nói trên khỏi đồ thị kết quả.

5. Các dạng chuẩn của các quan hệ

a) Khuyết tật của các lược đồ quan hệ

Khi thiết kế một hệ thống thông tin với mô hình quan hệ, người thiết kế ban đầu có thể tùy tiện gom nhóm các thuộc tính thành các lược đồ quan hệ theo nhận thức chủ quan của mình đối với thực tế. Các lược đồ quan hệ như vậy thường ẩn chứa một số biểu hiện chưa tốt, gọi là các khuyết tật. Các khuyết tật này lại thường liên quan tới các phụ thuộc hàm giữa các thuộc tính của lược đồ.

Sau đây ta minh họa các loại khuyết tật trên quan hệ CHẤMCÔNG cho ở Hình V.2.

- *Sự dư thừa thông tin:* Đó là sự lặp lại cùng một thông tin nhiều lần, ở nhiều chỗ. Nguồn gốc là ở sự có mặt của một phụ thuộc hàm $A \rightarrow B$, mà trong đó A không phải là khóa. Bởi vì theo định nghĩa của phụ thuộc hàm, lúc đó có khả năng có một cặp giá trị (a, b) của cặp thuộc tính <A, B> xuất hiện trên nhiều dòng của quan hệ, và điều này gây nên sự dư thừa thông tin. Trên quan hệ CHẤMCÔNG thì bởi có SH - Máy → SH - PX, cho nên cặp giá trị (m2, p_1), với nghĩa là: máy m2 đặt trong phân xưởng p1, đã được lặp 2 lần. Cũng tương tự, bởi có SH - PX → Trưởng - PX, nên cặp giá trị (p1, Giáp) với nghĩa là: phân xưởng p1 có trưởng phân xưởng là Giáp, lặp lại 3 lần; còn cặp (p2, Ất) được lặp 2 lần.

Chú ý rằng khi thuộc tính A trong phụ thuộc hàm $A \rightarrow B$ là khóa của quan hệ, thì không thể có sự dư thừa như trên, bởi vì A không thể lấy cùng một giá

trị a trên nhiều dòng khác nhau của quan hệ. Vậy có thể nói rằng loại dư thừa như trên sẽ không có nếu trong quan hệ, ngoài các phụ thuộc hàm giữa khóa và các thuộc tính ngoài khóa, thì không còn phụ thuộc hàm nào khác. Nói rõ hơn, hai loại phụ thuộc hàm sau đây, nếu có mặt, sẽ gây ra dư thừa thông tin:

+ PTH A→B, trong đó A là bộ phận của khóa, B ở ngoài khóa;

+ PTH A→B trong đó A và B đều ở ngoài khóa.

Ngoài sự dư thừa, thì khuyết tật của quan hệ còn thường lộ ra khi cập nhật, bổ sung hay loại bỏ bộ trong quan hệ.

- *Khuyết tật khi cập nhật:* Khi muốn điều chỉnh một thông tin, mà thông tin đó lại xuất hiện trên nhiều chỗ của quan hệ, thì ta phải tìm kiếm chúng, gây mất thời gian. Chẳng hạn muốn đổi (p1, Giáp) thành (p1, Mùi) thì phải duyệt toàn bộ các bộ của quan hệ và chữa ở 3 chỗ.

- *Khuyết tật khi loại bỏ một bộ:* Khi loại bỏ một bộ của quan hệ, có khả năng ta sẽ đánh mất một thông tin phụ (thể hiện bằng một phụ thuộc hàm) chứa trong bộ đó. Chẳng hạn khi công nhân C3 chết, đương nhiên là ta không cần chấm công cho anh ta nữa, cho nên ta sẽ loại các dòng thứ tư và thứ năm ra khỏi quan hệ CHẤMCÔNG; nhưng sau đó thì thông tin (p3, Đinh) sẽ không còn ở trong cơ sở dữ liệu nữa.

- *Khuyết tật khi bổ sung thông tin:* Khi ta muốn bổ sung một thông tin phụ chỉ chiếm một phần của một bộ, mà các giá trị khác trong bộ lại chưa có, thì ta chưa thể bổ sung được. Chẳng hạn, khi nhà máy vừa có thêm một phân xưởng mới là p4, với trưởng phân xưởng là Bính, thì ta cần bổ sung cặp (p4, Bính) vào cơ sở dữ liệu. Song điều đó là chưa làm được ngay, mà còn phải đợi cho tới khi có một công nhân làm việc thực sự ở phân xưởng mới đó và được chấm công thì lúc đó thông tin về phân xưởng này mới được bổ sung cùng với thông tin chấm công đối với công nhân nói trên.

b) Định nghĩa các dạng chuẩn

- Một lược đồ quan hệ R là ở *dạng chuẩn 1* (1NF) nếu các miền thuộc tính của nó đều là các miền đơn (nghĩa là không cấu thành từ nhiều miền khác). Thực ra yêu cầu miền đơn đã bao gồm trong định nghĩa quan hệ do Codd đưa ra năm 1970. Tuy nhiên sau đó năm 1980, Codd lại đưa thêm khái niệm quan hệ với miền thuộc tính không đơn (không chuẩn 1); ở đây ta không đề cập khái niệm mở rộng này.

- Một lược đồ quan hệ R là ở *dạng chuẩn 2* (2NF) nếu nó là 1NF và các phụ thuộc hàm giữa khóa và mỗi thuộc tính ngoài khóa đều là phụ thuộc hàm sơ đẳng. Nói cách khác, mọi thuộc tính ngoài khóa đều không phụ thuộc bộ phận vào khóa.

- Một lược đồ quan hệ R là ở *dạng chuẩn 3* (3NF) nếu nó là 2NF và các phụ thuộc hàm giữa khóa và mỗi thuộc tính ngoài khóa đều là phụ thuộc hàm trực tiếp. Nói cách khác là không tồn tại phụ thuộc hàm giữa các thuộc tính ngoài khóa.

Từ định nghĩa trên ta suy ra là trong một lược đồ quan hệ ở dạng chuẩn 3, thì ngoài các phụ thuộc hàm giữa khóa với các thuộc tính ngoài khóa, không còn phụ thuộc hàm nào khác. Như vậy nó sẽ tránh được các dạng khuyết tật mà ta đã xét ở trên. Tuy nhiên, những nghiên cứu sâu hơn cho thấy lược đồ quan hệ ở dạng chuẩn 3 vẫn còn có thể chứa đựng những loại khuyết tật khác. Chính vì vậy mà người ta còn đề xuất thêm nhiều dạng chuẩn khác nữa. Tuy nhiên trên thực tế thì xây dựng được một cơ sở dữ liệu gồm các quan hệ ở dạng chuẩn 3 đã là tốt lắm rồi, ta thường không đòi hỏi cao hơn.

c) Chuẩn hóa

Chuẩn hóa là sự phân rã (không làm mất mát thông tin) một quan hệ R thành một tập hợp các quan hệ ở dạng chuẩn 3.

Có nhiều giải thuật chuẩn hóa, nhưng tập trung theo hai hướng: phân tích và tổng hợp.

Không đi sâu vào lý thuyết, ở đây ta nêu ra hai cách làm khá đơn giản theo hai hướng khác nhau nói trên.

- *Chuẩn hóa theo hướng phân tích*

Thực hiện chuẩn hóa dần dần theo ba bước : 1NF, 2NF, rồi 3NF.

(i) Đưa về dạng chuẩn 1: Tách các thuộc tính lặp (không đơn)

+ Nhóm các thuộc tính đơn (còn lại) tạo thành một quan hệ. Chọn khóa cho nó.

+ Nhóm các thuộc tính lặp tách ra, tăng thêm khóa của quan hệ trên tạo thành một quan hệ (hay một số quan hệ theo chủ đề). Chọn khóa cho (các) quan hệ này, thường là khóa bội, trong đó khóa của quan hệ trên là một thành phần.

Các quan hệ lập được đều là 1NF.

(ii) Đưa về dạng chuẩn 2: Tách các nhóm thuộc tính phụ thuộc hàm vào một phần của khóa.

+ Nhóm còn lại tạo thành một quan hệ với khóa như cũ.

+ Mỗi nhóm tách ra (gồm các thuộc tính cùng phụ thuộc vào một (hay một số) thuộc tính nào đó của khóa) tăng thêm (các) thuộc tính mà chúng phụ thuộc tạo thành một quan hệ, với khóa là (các) thuộc tính tăng thêm này.

Các quan hệ lập được đều là 2NF.

(iii) Đưa về dạng chuẩn 3: Tách các nhóm thuộc tính phụ thuộc hàm vào một (hay một số) thuộc tính ngoài khóa.

+ Nhóm còn lại tạo thành một quan hệ với khóa như cũ.

+ Mỗi nhóm tách ra (gồm các thuộc tính cùng phụ thuộc vào một (hay một số) thuộc tính ngoài khóa) tăng thêm (các) thuộc tính mà chúng phụ thuộc, tạo thành một quan hệ, với khóa là (các) thuộc tính tăng thêm này.

Các quan hệ lập được đều là 3NF.

Thí dụ:

Thí dụ đầu tiên là một đơn hàng, trên đó ta gom được một danh sách các thuộc tính như sau:

- Các thuộc tính đơn:

  SH - ĐH

  SH - NgCCấp

  Tên - NgCCấp

  Địa chỉ - NgCCấp

  Ngày - ĐH

  Tổng cộng

- Các thuộc tính lặp (trong bảng của đơn hàng):

  Mã - MH

  Mô tả - MH

  Đơn vị tính

  Đơn giá

  Số lượng đặt

  Thành tiền

Các thuộc tính Thành tiền và Tổng cộng là các thuộc tính tính toán, bị loại khỏi danh sách.

Các phụ thuộc hàm ở trong danh sách thuộc tính còn lại là:

SH - ĐH → SH-NgCCấp, Tên - NgCCấp, Địa chỉ NgCCấp, Ngày ĐH

SH - ĐH, Mã - MH → Mô tả - MH, Đơn vị tính, Đơn giá, Số lượng đặt

SH - NgCCấp → Tên - NgCCấp, Địa chỉ NgCCấp

Mã - MH → Mô tả - MH, Đơn vị tính, Đơn giá

Các bước chuẩn hóa 1NF, 2NF, 3NF thực hiện như trong bảng cho ở Hình V.14, trong đó ta phân rã dần dần các quan hệ, khi đi từ cột này qua cột khác, từ trái sang phải. Một mũi tên diễn tả rằng quan hệ cũ được chuyển y nguyên sang cột mới.

Cuối cùng ta thu được 4 quan hệ (ở 3NF). Đặt tên cho các quan hệ đó (căn cứ vào khóa của chúng), ta có:

Đơn hàng (SH - ĐH, SH - NgCCấp, Ngày ĐH)

NgCCấp (SH - NgCCấp, Tên - NgCCấp, Địa chỉ - NgCCấp)

Dòng ĐH (SH - ĐH, Mã - MH, Số lượng đặt)

Mặt hàng (Mã - MH, Mô tả- MH, Đơn vị tính, Đơn giá)

Chứng từ giao dịch: ĐƠN HÀNG			
Danh sách thuộc tính	1NF	2NF	3NF
SH - ĐH SH - NgCCấp Tên - NgCCấp Địa chỉ - NgCCấp Ngày - ĐH Mã - MH Mô tả - MH Đơn vị tính Đơn giá Số lượng đặt	SH - ĐH SH - NgCCấp Tên - NgCCấp Địa chỉ - NgCCấp Ngày - ĐH SH - ĐH Mã - MH Mô tả - MH Đơn vị tính Đơn giá Số lượng đặt	→ SH - ĐH Mã - MH Số lượng đặt Mã - MH Mô tả - MH Đơn vị tính Đơn giá	SH - ĐH SH - NgCCấp Ngày - ĐH SH - NgCCấp Tên - NgCCấp Địa chỉ - NgCCấp → →

Hình V.14 Chuẩn hóa một đơn hàng

Thí dụ thứ hai là quan hệ CHẤMCÔNG, cho ở Hình V.2. Ở đây quan hệ đã ở dạng chuẩn 1, ta chỉ cần thực hiện hai bước 2NF và 3NF (xem Hình V.15). Đặt tên cho các quan hệ 3NF thu được, ta có kết quả chuẩn hóa sau:

Công (Mã - CN, SH - máy, Thời gian)

Máy (SH - máy, SH - PX)

Phân xưởng (SH - PX, TrưởngPX)

Sổ ghi chép: CHẤM CÔNG			
Danh sách thuộc tính	1NF	2NF	3NF
Mã - CN SH - Máy Thời gian SH - PX Trưởng PX	Mã - CN SH - Máy Thời gian SH - PX Trưởng PX	Mã - CN SH - Máy Thời gian SH - Máy SH - PX Trưởng PX	→ SH - Máy SH - PX SH - PX Trưởng PX

Hình V.15 Chuẩn hóa bảng chấm công

Nếu viết đầy đủ giá trị các thuộc tính, ta có 3 bảng giá trị như sau:

Công:

Mã - CN	SH - Máy	Thời gian
c1	m1	10
c1	m2	10
c2	m3	50
c3	m5	100
c3	m4	30
c2	m2	20

Máy:

SH - Máy	SH - PX
m1	p1
m2	p1
m3	p2
m5	p3
m4	p2

Phân xưởng:

SH - PX	Trưởng - PX
p1	Giáp
p2	Ất
p3	Đinh

Thực hiện phép kết nối đối với 3 bảng này, ta thu lại được bảng ở Hình V.2, nghĩa là sự phân rã đã không làm mất mát thông tin.

Mặt khác, các khuyết tật nêu ở mục 5a ở trên cũng không còn nữa:

- Các sự lặp lại thông tin như (p1, Giáp) lặp 3 lần, thì nay không còn.
- Khi cập nhật (p1, Giáp) thành (p1, Mùi) thì chỉ cần điều chỉnh một chỗ (ở quan hệ Phân xưởng) mà không phải là ở 3 chỗ như cũ.
- Khi bổ sung thông tin (p4, Bính), chỉ việc thêm một bộ mới cho quan hệ Phân xưởng (mà không cần chờ đợi một sự chấm công).
- Khi loại bỏ các thông tin về công nhân C3 (đã chết) chỉ việc loại các dòng 4 và 5 trong quan hệ Công, mà không làm đánh mất thông tin (p3, Đinh) nằm ở quan hệ Phân xưởng.

• *Chuẩn hóa theo hướng tổng hợp*

Cách làm này cho ngay các lược đồ quan hệ 3NF mà không qua các giai đoạn 1NF, 2NF.

(i) Xuất phát từ một danh sách các thuộc tính. Tìm các phụ thuộc hàm giữa các thuộc tính trong danh sách đó.

(ii) Lập đồ thị các phụ thuộc hàm (xem mục 4a):

- Mỗi thuộc tính trong danh sách là một nút.
- Mỗi nhóm thuộc tính là vế trái một PTH cũng là một nút.
- Nếu có PTH A → B thì vẽ một cung nối nút A tới nút B.

(iii) Đưa đồ thị về phủ tối tiểu của nó:

- Hoặc làm bằng tay (xem mục 4b), mà biến đổi chủ yếu là loại bỏ các cung khép kín hình tam giác.
- Hoặc làm bằng máy tính(với cặp giải thuật cho ở mục 4c).

Tới đây trên đồ thị chỉ còn các PTH trực tiếp.

(iv) Dùng các hình chữ nhật (hay hình thang) để khoanh vùng trên đồ thị thành các quan hệ như sau: mỗi nút trong (tức là nút có con) lấy làm khóa, hợp cùng với các con của nó lập thành một quan hệ.

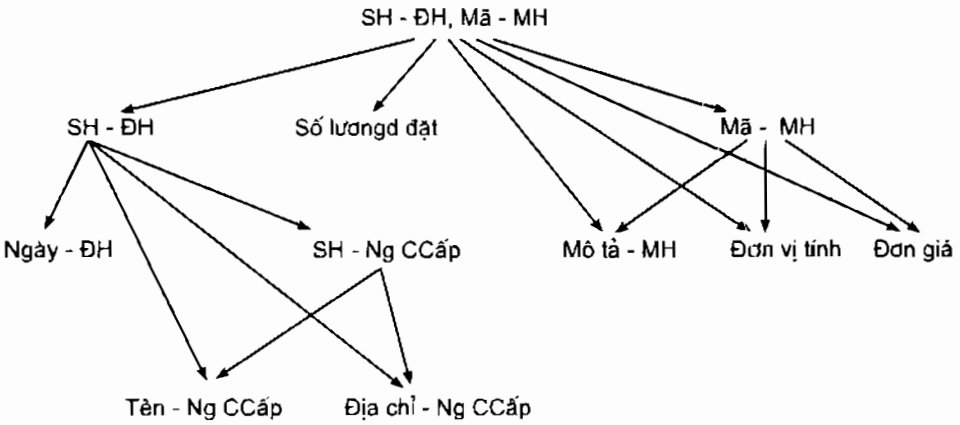
Các quan hệ lập được đều là 3NF.

Thí dụ:

Trở lại thí dụ Đơn hàng ở trên, với danh sách các PTH đã liệt kê, thêm các PTH tầm thường:

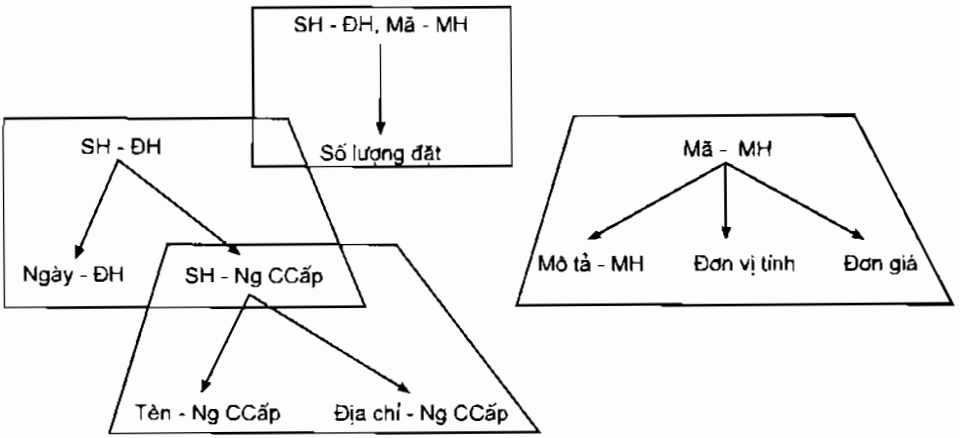
SH - ĐH, Mã - MH → SH - ĐH, Mã - MH

ta lập được đồ thị PTH như ở Hình V.16.



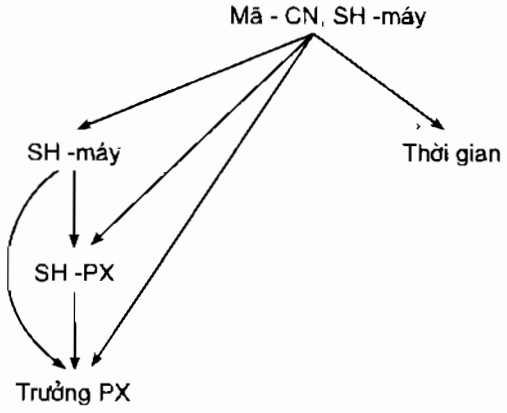
Hình V.16 Đồ thị PTH lập từ một Đơn hàng

Từ đồ thị đó, sau khi loại bỏ các cung khép kín hình tam giác, và loại bỏ nốt hai PTH tầm thường đã được đưa vào trên kia, ta thu được đồ thị phủ tối tiểu, trong đó có 4 nút trong tương ứng với 4 quan hệ 3NF như ta đã có được từ phương pháp trên (Hình V.17).

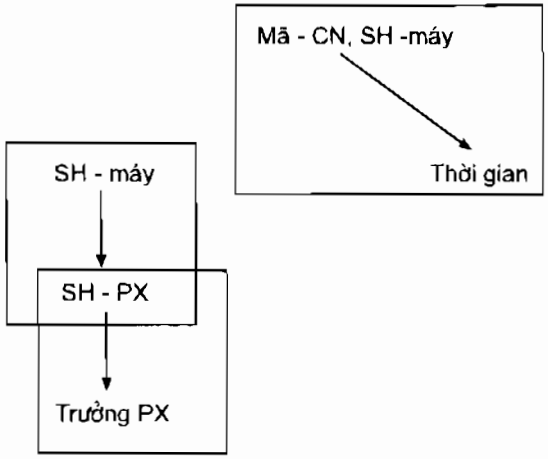


Hình V.17 Phủ tối tiểu và các quan hệ 3NF từ Đơn hàng

Chuyển sang thí dụ Chấmcông, ta cũng có tương tự: đầu tiên là đồ thị các PTH như trong Hình V.18, rồi phủ tối tiểu và các quan hệ 3NF thu được, như trong Hình V.19.



Hình V.18 Đồ thị TPH từ bảng chấm công



Hình V.19 Phủ tối tiểu và các quan hệ 3NF từ bảng Chấm công

6. Phương pháp lập lược đồ dữ liệu theo mô hình quan hệ

Sau khi đã giải quyết các vấn đề "kỹ thuật" (đặc biệt là về sự chuẩn hóa), ta có thể tóm tắt quy trình thành lập lược đồ dữ liệu cho hệ thống, theo mô hình quan hệ là như sau:

(i) Thành lập một danh sách các thuộc tính, gọi là *danh sách xuất phát*. Có thể xem đây là một quan hệ, với một ý nghĩa khái quát nào đó. Không ngại rằng danh sách này không bao trùm được các dữ liệu của toàn hệ thống, bởi vì

quá trình sẽ còn được lặp lại với nhiều danh sách xuất phát khác cho đến khi vét cạn các thông tin cần thiết cho hệ thống.

Có hai cách tiếp cận cho việc thành lập danh sách xuất phát:

- Cách thứ nhất: Đó là tập hợp các thông tin cơ bản (thông tin không có cấu trúc) phát hiện được trong một phạm vi điều tra nào đó, mà ta xem là đáng ghi nhận (nghĩa là có ích cho việc quản lý). Nếu đã có một bước mô hình hóa sơ bộ với mô hình Thực thể/Liên kết, thì đó là tập hợp các thuộc tính của một số kiểu thực thể hay kiểu liên kết nào đó.

- Cách thứ hai: Xuất phát từ một hay một số cái ra của hệ thống. Cái ra có thể là:

  + một chứng từ hay một tài liệu in ra từ hệ thống

  + màn hình trong giao tiếp người/máy.

Danh sách xuất phát sẽ gồm mọi tiêu thức (tên các thông tin) xuất hiện trên các cái ra đó.

(ii) Tu sửa lại danh sách xuất phát, qua các công việc sau:

- Loại bỏ bớt các tên đồng nghĩa.

- Loại bỏ các thuộc tính tính toán, tức là các thuộc tính có giá trị được tính từ giá trị của những thuộc tính khác.

  Thí dụ: Thành tiền = Đơn giá × Số lượng

  Tổng cộng = Σ Thành tiền

- Truy nguyên các thuộc tính dùng để tính toán các thuộc tính đã bị loại trên, nếu chúng chưa có mặt trong danh sách thì kết nạp vào. Chính nhờ sự truy nguyên này mà phương pháp xuất phát từ cái ra rốt cuộc vẫn thu nạp được các thông tin cần thiết. Tuy nhiên như vậy, hệ thống cũng chỉ có vừa đủ các thông tin để sản sinh các cái ra đã được dự kiến sẵn mà thôi.

- Nếu có thể được, thay thế các thuộc tính không đơn (có giá trị là dãy giá trị hay tập hợp giá trị) bởi các thuộc tính đơn. Chẳng hạn trong hồ sơ ghi nhận kết quả học tập trong năm học của một sinh viên có thuộc tính Điểm Toán. Thực chất đấy là một dãy ba số: kết quả thi lần 1, lần 2 và lần 3 của môn Toán (chưa thi hay không thi lần thi nào, thì lấy giá trị Null - một giá trị đặc biệt - cho lần đó). Vậy có thể thay thuộc tính Điểm

Toán bằng ba thuộc tính đơn là: Điểm Toán 1, Điểm Toán 2, Điểm Toán 3. Việc thay các thuộc tính không đơn bởi các thuộc tính đơn không phải bao giờ cũng làm được, chẳng hạn khi số các giá trị hợp thành là không cố định. Các thuộc tính không đơn sót lại, khi chuyển sang bước tiếp sau sẽ được xem là thuộc tính đơn để tìm PTH cho nó.

(iii) Tìm các phụ thuộc hàm trong danh sách các thuộc tính:

- Đầu tiên là rà các khả năng có PTH giữa từng cặp các thuộc tính trong danh sách.

- Sau đó xét các PTH có vế trái gồm 2, 3,... thuộc tính.

Các PTH tầm thường thì có thể bỏ qua, song các PTH phản ánh một thực tế quản lý, thì nhất thiết không được bỏ sót. Như trên đã nói, khi tìm PTH thì các thuộc tính trong danh sách, dù trước đó vốn là thuộc tính không đơn, đều được xem là thuộc tính đơn. Chẳng hạn nếu dùng ba thuộc tính Điểm Toán 1, Điểm Toán 2, Điểm Toán 3 cho 3 điểm Toán ở ba lần thi, thì có các PTH:

Mã -SV → Điểm Toán 1

Mã -SV → Điểm Toán 2

Mã -SV → Điểm Toán 3

Tuy nhiên nếu giữ lại thuộc tính Điểm Toán và xem nó là đơn, thì lại có PTH:

Mã -SV, Lần thi → Điểm Toán

Các thuộc tính trỏ thời gian hay không gian (như thuộc tính Lần thi) được gọi là thuộc tính Không - Thời gian.

Khi cần xuất hiện trong một phụ thuộc hàm, thì một thuộc tính không - thời gian có thể được bổ sung vào danh sách các thuộc tính đang xét.

(iv) Tiến hành chuẩn hóa dựa trên tập các PTH đã lập được ở trên, sử dụng một trong các phương pháp chuẩn hóa đã biết. Kết quả thu được là một tập các lược đồ quan hệ 3NF.

(v) Lặp lại các bước từ (i) đến (iv) cho các danh sách xuất phát khác nhau, cho đến khi quét hết các phạm vi khảo sát. Ta được nhiều tập lược đồ quan hệ 3NF.

(vi) Lấy hợp các kết quả thu được từ các lần lặp trên. Khi lấy hợp như vậy, nếu có hai quan hệ có khóa trùng nhau, thì chúng được gộp thành một quan hệ

với danh sách các thuộc tính là hợp của hai danh sách tương ứng. Chẳng hạn, từ Đơn hàng, ta thu được quan hệ:

Khách hàng (SH-KH, Tên -KH, Địa chỉ-KH)

và từ Phiếu đăng ký khách hàng, ta lại thu được quan hệ:

Khách hàng (SH-KH, Tên -KH, Giới hạn vay nợ, Tình trạng)

thì ta phải gộp hai quan hệ đó làm một, thành:

Khách hàng (SH-KH, Tên -KH, Địa chỉ-KH, Giới hạn vay nợ, Tình trạng).

Có thể khi gom hai danh sách thuộc tính như vậy, ta có thể làm xuất hiện PTH mới trong danh sách làm cho quan hệ lập được không còn ở dạng chuẩn ba, lúc đó ta lại phải chuẩn hóa lại cho quan hệ đó.

Tóm tắt

Tới đây ta đã trình bày hai mô hình dữ liệu: mô hình thực thể/liên kết (chương IV) và mô hình quan hệ (chương V). Mô hình thực thể/liên kết dễ dùng hơn và mô hình quan hệ chặt chẽ hơn. Chính vì vậy mô hình thực thể/liên kết thường được vận dụng trong bước đầu lập mô hình, còn mô hình quan hệ được vận dụng trong bước hoàn chỉnh mô hình.

Ba mức của mô hình thực thể/liên kết (mở rộng, kinh điển và hạn chế) thu hẹp dần các hình thức biểu diễn. Trong hai mô hình mở rộng và kinh điển, ta dùng các nút của một đồ thị để diễn tả các kiểu thực thể, và các cung trong của đồ thị để diễn các mối liên quan giữa chúng (như sự thừa kế, sự liên kết). Đó là một cách diễn tả rất tự nhiên, giúp ta dễ vận dụng mô hình để phản ánh thực tế. Nhưng qua mô hình hạn chế thì đồ thị chỉ còn nút mà không còn cung (đúng ra là có vẽ cung bằng các đường nối 1- nhiều, song chúng có thể bị loại bỏ hẳn), vì các liên kết đã "lặn" vào các thuộc tính (với sự có mặt của các thuộc tính kết nối). Như vậy mô hình hạn chế có vẻ ẩn dụ hơn.

Nhưng ưu điểm của mô hình hạn chế là ở chỗ nó có thể được chuyển đổi trực tiếp sang mô hình quan hệ: mỗi kiểu thực thể tương ứng 1-1 với một lược đồ quan hệ. Nhờ vậy mô hình có thể được chuẩn hóa một cách dễ dàng.

Người dùng còn ít kinh nghiệm nên dùng các mô hình TT/LK mở rộng hay kinh điển để phản ánh thực tế. Sau đó dùng các quy tắc 1-6 để chuyển sang mô hình hạn chế, từ đó sang mô hình quan hệ rồi thực hiện chuẩn hóa.

Người dùng có kinh nghiệm thì có thể bắt đầu ngay từ mô hình TT/LK hạn chế, thậm chí có thể không dùng mô hình TT/LK nữa, mà bắt đầu với một danh sách các thuộc tính, thu gom từ một phạm vi điều tra hoặc từ các cái ra của hệ thống, xem đó như là một quan hệ xuất phát không chuẩn, rồi chuẩn hóa, và lặp lại việc đó nhiều lần để có được mô hình dữ liệu hoàn chỉnh.

§2. CÁC RÀNG BUỘC TOÀN VẸN

Các mô hình dữ liệu, dù là mô hình thực thể/liên kết hay mô hình quan hệ, vẫn không đủ để diễn tả mọi ngữ nghĩa của thế giới thực mà ta cần phải mô hình hóa trong hệ thống thông tin. Bởi vậy ta phải bổ sung thêm các đặc tả, thường gọi là các *ràng buộc toàn vẹn*.

Ràng buộc toàn vẹn (integrity constraint) là một đặc tả mà một cơ sở dữ liệu phải thỏa mãn để giữ được tính đúng đắn (toàn vẹn) của nó. Ràng buộc toàn vẹn thường được diễn đạt dưới dạng một tân từ.

Các ràng buộc toàn vẹn gắn kết với một mô hình quan hệ thường chia thành hai loại: Các ràng buộc toàn vẹn tĩnh và các ràng buộc toàn vẹn động. Mỗi loại đó lại gồm nhiều loại nhỏ mà ta sẽ lần lượt đề cập sau đây.

1. Các ràng buộc toàn vẹn tĩnh

Ràng buộc toàn vẹn tĩnh là một điều kiện mà cơ sở dữ liệu phải thỏa mãn vào bất cứ lúc nào, để có thể ở trong trạng thái đúng đắn. Có bốn loại ràng buộc toàn vẹn tĩnh:

a) Các ràng buộc trên thuộc tính

Bao gồm:

- *Ràng buộc tồn tại*, gắn kết với một thuộc tính của một lược đồ quan hệ, nó đòi hỏi thuộc tính đó phải xác định trong mọi bộ-n của quan hệ, nghĩa là phải có một giá trị khác với giá trị không xác định (NULL). Bình thường thì thuộc tính khóa của một quan hệ phải chịu ràng buộc này.
- *Ràng buộc về miền* định nghĩa một cách chính xác miền các giá trị của một thuộc tính. Định nghĩa đó có thể là danh sách các giá trị được phép,

có thể là một khoảng các giá trị, và cũng có thể cho theo một khuôn dạng đặc biệt nào đó.

Thí dụ:

- Thuộc tính TTGIADINH, cho biết tình trạng gia đình của một nhân viên, lấy các giá trị trong danh sách: (COV/C, DOCTHAN, LYDI, GOA).
- Thuộc tính TUỔI của học sinh cấp II lấy các giá trị trong [1..18].
- Thuộc tính SO-BAOHIEM luôn bắt đầu bởi số 1 hoặc số 2.

• *Ràng buộc về giá trị mặc định* chỉ rõ giá trị mà ta phải gán cho một thuộc tính khi tạo lập một bộ-n, nếu lệnh tạo lập không gán giá trị cho thuộc tính đó.

Chẳng hạn thuộc tính NGAY-HĐ lấy giá trị mặc định là ngày hiện tại.

b) Các ràng buộc trên bộ - n

Ràng buộc này thể hiện bằng một tân từ hay một công thức đề cập đến các giá trị của nhiều thuộc tính của một bộ-n.

Thí dụ:

- Trong bảng NHÂNSỰ, có thể có một ràng buộc trên bộ-n là: Nếu SO-BAOHIEM (số bảo hiểm của một người) bắt đầu bởi số 1 thì thuộc tính GIOI (giới tính) phải lấy giá trị NAM và trong trường hợp ngược lại, thì lấy giá trị NU.
- Trong bảng DONGDONHANG, có thể có ràng buộc trên bộ-n là: Giá trị của thuộc tính THANHTIEN là bằng tích các giá trị của các thuộc tính SOLUONG và DONGIA..

c) Ràng buộc trên quan hệ

Có nhiều loại ràng buộc trên quan hệ.

• *Ràng buộc về khóa*, chỉ ra một hay một số thuộc tính nào đó là khóa. Nó tạo ra một ràng buộc trên tập hợp các bộ-n của quan hệ.

• *Ràng buộc phụ thuộc hàm*, chỉ ra rằng có một phụ thuộc hàm giữa hai thuộc tính hay hai nhóm thuộc tính. Nó đòi hỏi mọi quan hệ trong một lược đồ quan hệ phải tuân thủ phụ thuộc hàm đó.

• *Ràng buộc về dạng chuẩn* đối với một lược đồ quan hệ là một đòi hỏi rằng mọi quan hệ trong lược đồ đó đều phải ở dạng chuẩn đã được chỉ ra.

- *Ràng buộc về bản số* đối với một lược đồ quan hệ đưa ra các cận cho số tất cả các bộ-n trong mỗi quan hệ thuộc lược đồ, hoặc cho số các bộ-n trong quan hệ thỏa mãn một tính chất nào đó.

Thí dụ: Trong quan hệ DONGDONHANG có thể có ràng buộc là: mỗi đơn hàng không có quá 8 dòng đơn hàng.

d) Ràng buộc trên nhiều quan hệ

Cũng có nhiều loại:

- *Ràng buộc về khóa ngoài*, chỉ ra rằng một thuộc tính B của một quan hệ R1 là cùng ngữ nghĩa với một thuộc tính A, là khóa chính của một quan hệ R2. Bấy giờ B được gọi là khóa ngoài (hay thuộc tính kết nối) của R1 tham chiếu tới R2. Lưu ý rằng, thông thường thì A và B có cùng tên trong hai quan hệ R1 và R2, song điều này là không bắt buộc.

Thí dụ:

KHACHHANG (SH-KH, TEN-KH) khóa chính: SH - KH.

DONHANG (SH-ĐH, SH-KH, ...) khóa chính: SH-ĐH;

Khoá ngoài: SH-KH tham chiếu tới KHACHHANG

SANPHAM (SH-SP,...) khóa chính: SH-SP

HOPTHANH (SH-TOHOP, SH-THANHPHAN, SLG)

Khoá chính: SH-TOHOP, SH-THANHPHAN;

Khóa ngoài: SH-TOHOP tham chiếu SANPHAM;

Khóa ngoài: SH-THANHPHAN tham chiếu SANPHAM.

- *Ràng buộc toàn vẹn tham chiếu*: đòi hỏi các giá trị được thực nhận của các khóa ngoài phải bao hàm trong các giá trị của các khóa chính tương ứng. Nói rõ hơn là: Giả sử R2 (A,...) là một lược đồ quan hệ có khóa chính là A và R1 (..., A,...) là một lược đồ quan hệ với A là khóa ngoài tham chiếu tới R1, thì: Trong mọi bộ-n r1 của R1, giá trị của thuộc tính A, tức là r1.A, phải thuộc vào chiếu của quan hệ R2 trên A, có nghĩa là tồn tại một bộ-n r2 của R2 sao cho r2. A = r1.A.

Thí dụ: Trở lại hai thí dụ Đơn hàng và Hợp thành ở trên, thì ràng buộc toàn vẹn tham chiếu có nghĩa là:

- Không thể có một giá trị của SH-KH trong bảng DONHANG mà lại không có trong bảng KHACHHANG. Điều đó cũng có nghĩa là không thể ghi nhận một Đơn hàng cho một khách hàng không tham chiếu được.

- Cũng tương tự, không thể mô tả sự hợp thành giữa hai sản phẩm chỉ ra trong các thuộc tính SH-TOHOP và SH-THANHPHAN mà các sản phẩm này lại không có trong bảng SANPHAM.

• Cũng còn nhiều *ràng buộc riêng biệt* khác trên nhiều quan hệ còn chưa được xếp loại trong mô hình quan hệ. Chúng thường được diễn tả dưới dạng các khẳng định, hoặc là bằng ngôn ngữ tự nhiên, hoặc là bằng ngôn ngữ hình thức xây dựng trên lôgic các tân từ.

2. Các ràng buộc toàn vẹn động

Đó là các ràng buộc toàn vẹn nhằm đặc tả một thay đổi trạng thái đúng đắn của cơ sở dữ liệu. Phân biệt hai loại:

a) Các ràng buộc định nghĩa bởi các tiên đề, hậu đề

Các ràng buộc này diễn tả các điều kiện mà cơ sở dữ liệu cần phải nghiệm đúng khi tiến hóa. Đó có thể là:

• *Ràng buộc loại tiền đề*: chỉ ra một điều kiện phải thỏa mãn TRƯỚC khi tiến hóa;

• *Ràng buộc loại hậu đề*: Chỉ ra một điều kiện phải thỏa mãn SAU khi tiến hóa;

• *Ràng buộc tiền và hậu đề*: Chỉ ra một điều kiện so sánh các trạng thái của cơ sở dữ liệu TRƯỚC và SAU tiến hóa.

Thí dụ:

- Không được loại bỏ một khách hàng còn có các đơn hàng chưa thanh toán xong. Điều đó có thể diễn tả như là một ràng buộc loại tiền đề cho thao tác loại bỏ trong bảng KHACHHANG.

- Một nhân viên không thể làm quá 50 giờ vượt trội trong một tháng. Điều đó có thể diễn tả như là một ràng buộc hậu đề cho thao tác bổ sung trong bảng GIO-VUOT.

- Khi cập nhật lượng hàng đã phát cho một dòng đơn hàng, thì lượng hàng đó chỉ có thể tăng lên mà vẫn còn ở dưới lượng hàng đặt mua. Điều đó có thể diễn tả bằng một tân từ so sánh các giá trị của thuộc tính

LUONGDAPHAT TRƯỚC và SAU khi điều chỉnh một bộ-n của quan hệ DONGDONHANG, tức là một ràng buộc tiền và hậu đề.

b) Các ràng buộc định nghĩa bởi luật ứng xử

Các ràng buộc này chỉ ra các hành động phải thực hiện đáp lại một thay đổi trạng thái chợt đến trong cơ sở dữ liệu. Chúng thường có dạng: Khi một sự kiện thuộc một loại nào đó phát sinh và nếu một điều kiện nào đó nghiệm đúng, thì thực hiện một hành động nào đó.

Thí dụ: Khi khách hàng gửi tới một yêu cầu hủy đơn hàng, nếu chưa có dòng đơn hàng nào đã phát hàng đi, thì loại bỏ đơn hàng cùng với mọi dòng đơn hàng của nó. Ở đây sự kiện chợt đến là yêu cầu hủy đơn hàng của khách hàng, điều kiện là chưa có phần nào của đơn hàng đó đã được thực hiện, và hành động là loại bỏ bộ-n tương ứng trong quan hệ DONHANG cùng với các bộ-n liên quan trong quan hệ DONGDONHANG.

3. Phương thức biểu diễn các ràng buộc toàn vẹn

Không có một chuẩn thống nhất cho hình thức biểu diễn các ràng buộc toàn vẹn. Tuy nhiên ta có thể phỏng theo cach biểu diễn của hệ QTCSDL Sybase như sau đây:

Một ràng buộc toàn vẹn được định nghĩa bởi một *tên*, một *loại* (TINH, DONG), một *tầm* (THUOCTINH, QUANHE, CACQUANHE) và một *văn bản*. Đối với các ràng buộc động, còn phải chỉ rõ đó là ràng buộc *tiền, hậu,* hay *tiền-hậu*, và một luật *ứng xử*.

Vậy có thể biểu diễn các ràng buộc toàn vẹn theo các mẫu sau:

- ASSERT RANGBUOC TINH tên của ràng buộc TREN THUOCTINH tên thuộc tính CUA tên quan hệ: văn bản.
- ASSERT RANGBUOC TINH tên của ràng buộc TREN QUANHE tên của quan hệ: văn bản.
- ASSERT RANGBUOC TINH tên của ràng buộc GIUA CACQUANHE danh sách các tên quan hệ: văn bản.
- ASSERT RANGBUOC DONG tên của ràng buộc TRUOC/SAU/ TRUOC-SAU tên hành động: văn bản.

Nếu có thể được thì các văn bản trong các ràng buộc nên viết dưới dạng một tân từ của lôgic cấp một, kèm thêm một chú giải bằng ngôn ngữ tự nhiên.

Chương VI

PHÂN TÍCH HỆ THỐNG VỀ ĐỘNG THÁI

Như ta đã thấy thì phương pháp SA không mô hình hóa sắc thái điều khiển trong hệ thống, không đề cập tới các loại biến cố đột xuất, các xử lý không đồng bộ, cũng như các xử lý song hành. Biểu đồ luồng dữ liệu với nguyên tắc ngầm định là kích hoạt bởi dữ liệu, chỉ là một mô hình lý tưởng hóa, không tính đến những biến động trong thời gian của hành vi hệ thống.

Tuy nhiên, sự biến đổi hành vi qua thời gian mà ở đây ta gọi là động thái (dynamics), lại là một hiện tượng thường có trong các hệ thống thời gian thực, không thể bỏ qua được.

Chương này trình bày phương pháp SART (Structured Analysis For Real–Time Systems), do Hatley và Pirbhai đề xuất, nhằm phân tích động thái của hệ thống mà vẫn giữ xu hướng phân tích trên xuống. Các bạn đọc chỉ quan tâm các hệ thống thông tin quản lý, ít chú ý đến khía cạnh thời gian thực, thì có thể bỏ qua chương này.

§1. PHÂN TÍCH ĐỘNG THÁI VỚI CÁC BIỂU ĐỒ LUỒNG ĐIỀU KHIỂN

Cái mới mà SART đưa thêm vào để mô hình hóa động thái là biểu đồ luồng điều khiển, mà ta trình bày bước đầu trong tiết này và sẽ chi tiết hóa thêm trong tiết sau.

1. Khái niệm về luồng điều khiển

Trong một biểu đồ luồng dữ liệu (BDL), thì một luồng dữ liệu nối từ một chức năng A tới một chức năng B là một thông tin được chuyển giao từ A sang B, thông tin đó là một dữ liệu ra (tức là một sản phẩm xử lý) của A và là một dữ liệu vào (tức là một nguyên liệu cho xử lý) của B. Không thể hiểu nó là một thông tin điều khiển, chẳng hạn không thể hiểu như trong các sơ đồ khối (flow chart), là A chuyển điều khiển (tức là chuyển quyền thực hiện) cho B.

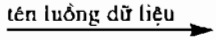
Nguyên tắc "kích hoạt bởi dữ liệu" cho phép hiểu là chức năng B sẽ được khởi động (và sẽ thực hiện vô cùng nhanh) khi nó hội đủ các dữ liệu đầu vào, chỉ là một sự khỏa lấp tạm thời sự trống vắng của các yếu tố điều khiển trong các BLD mà thôi.

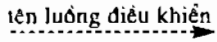
SART khắc phục thiếu sót này của BLD bằng cách đưa thêm khái niệm luồng điều khiển (control flow).

> *Luồng điều khiển* là một thông tin đến hoặc ra khỏi một chức năng (chức năng xử lý hay chức năng điều khiển), với tác dụng là làm thay đổi trạng thái nội tại của chức năng nhận nó (nếu chức năng này là chức năng điều khiển) hoặc làm thay đổi tình trạng thực hiện (khởi động, tạm ngưng, khôi phục, kết thúc) của chức năng nhận nó (nếu chức năng này là một chức năng xử lý).

Chức năng xử lý là chức năng biến đổi dữ liệu (thuộc lĩnh vực ứng dụng), còn chức năng điều khiển là chức năng có khả năng nhận và phát các luồng điều khiển nhằm khống chế quá trình thực hiện các phần của hệ thống.

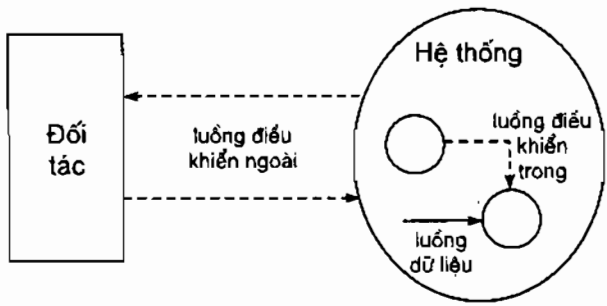
Luồng điều khiển trong SART được biểu diễn khác biệt với luồng dữ liệu :

- Luồng dữ liệu : 

- Luồng điều khiển : 

Nếu xét về nơi phát và nơi nhận, thì một luồng điều khiển có thể là (Hình VI.1):

- luồng điều khiển ngoài, nếu nơi phát hoặc nơi nhận là một đối tác (phần tử ngoài hệ thống);
- luồng điều khiển trong, nếu cả nơi phát và nơi nhận đều là những chức năng bên trong hệ thống.



Hình VI.1 Các luồng điều khiển ngoài và trong

Nếu xét về miền giá trị, thì một luồng điều khiển *luôn luôn* là một tín hiệu rời rạc, và *không bao giờ* là liên tục. Còn luồng dữ liệu thì *thường* là một thông tin liên tục (như tốc độ, độ cao, gia tốc, khoảng thời gian, áp suất, nhiệt độ v.v...), và *đôi khi* cũng có thể là rời rạc (chẳng hạn tháng, học hàm v.v...).

Nếu xét về nội dung mang tải, thì luồng điều khiển có thể là:

- một tín hiệu, tức là một thông tin mang tên, có tác dụng điều khiển, chẳng hạn lệnh "bắt đầu" từ thao tác viên, lệnh "tăng tốc" gửi tới một cơ chế điều khiển mô tơ, một cái nhấp chuột, một cái ấn phím v.v...
- một biến cố xảy ra đột xuất, chẳng hạn một cái ngắt, một sai lỗi v.v...
- một thay đổi:
  + thay đổi trạng thái (trạng thái là một giai đoạn trong đó không có sự thay đổi về hành vi, giữa hai lần biến động), chẳng hạn lò hơi đủ nóng, bình chứa đầy v.v...
  + thay đổi cung cách hoạt động (khởi động, tạm ngưng, khôi phục, kết thúc).
- một điều kiện Bun (gọi là điều kiện dữ liệu hay điều kiện đồng bộ hóa) được thỏa mãn, chẳng hạn "đã hết tháng và đã có bảng chấm công" (để khởi động việc tính lương).
- một thời hạn kết thúc, chẳng hạn "cuối học kỳ", "đã qua hai giây" v.v...

2. Biểu đồ luồng điều khiển và sự phân tích động thái

a) Phân tích trên xuống

Việc phân tích hệ thống về mặt động thái cũng được tiến hành trên xuống. Mô hình động thái là 1 tập hợp các biểu đồ luồng điều khiển (BLĐ) được phân mức và sóng đôi với các BLD.

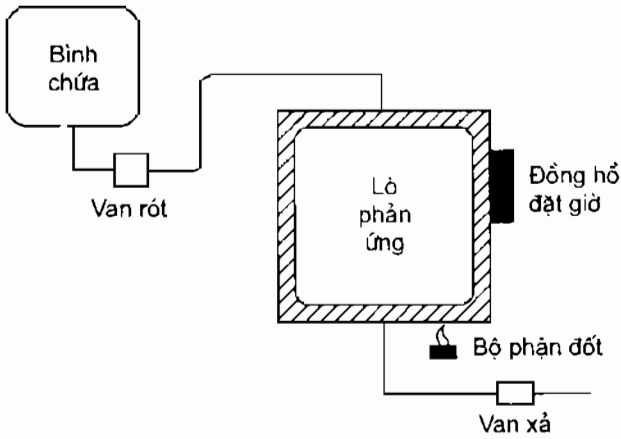
Mỗi BLĐ sử dụng *cùng các chức năng XL* với BLD sóng đôi với nó. Tuy nhiên nó chỉ rõ thêm các *luồng điều khiển* và các *chức năng điều khiển* xử lý các luồng điều khiển đó. Mục đích của BLĐ là mô tả các chức năng xử lý dữ liệu trong BLD tương ứng được khởi động, ngưng ngắt và đồng bộ hóa như thế nào.

Để trình bày rõ quá trình phân tích động thái từ trên xuống này, ta hãy xét một thí dụ sử dụng xuyên suốt cả chương này.

Thí dụ ĐK: Một hệ thống điều khiển phản ứng nhiệt (HĐPN). Một phản ứng nhiệt đòi hỏi phải duy trì một lượng nhất định một chất phản ứng lỏng ở một nhiệt độ D trong một khoảng thời gian T. Việc rót đầy lò phản ứng bắt đầu khi có lệnh "bắt đầu rót" từ thao tác viên. Khi chất phản ứng trong lò đã đạt một mức N thì ngừng rót và bật bộ phận đốt nóng. Khi nhiệt độ trong lò đạt tới

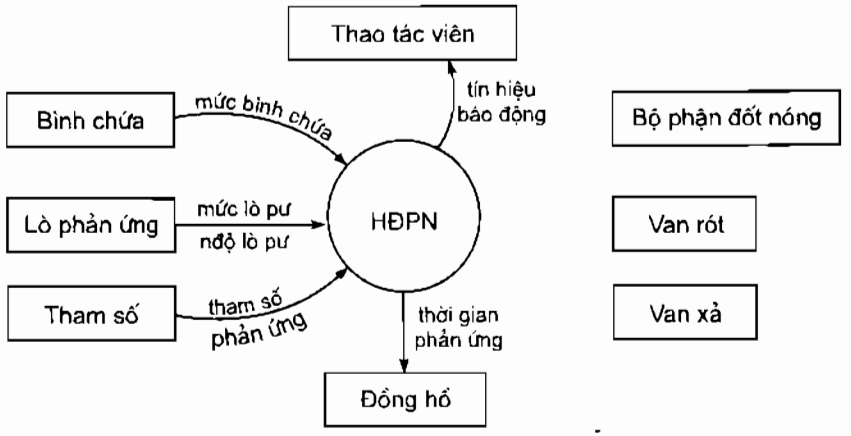
giá trị D thì đặt đồng hồ báo giờ (thời hạn = T). Duy trì nhiệt độ của phản ứng ở giá trị cố định D trong khoảng thời gian T. Khi đồng hồ báo hết thời hạn T thì tắt bộ phận đốt nóng và xả chất phản ứng ra một bể chứa bên ngoài.

Chú ý: Việc rót chất phản ứng vào lò chưa thể bắt đầu được nếu mức chất phản ứng trong bình chứa chưa đạt một giá trị tối thiểu M. Hệ thống phải phát hiện được điều này để đưa ra một báo động đáp lại lệnh "bắt đầu rót".

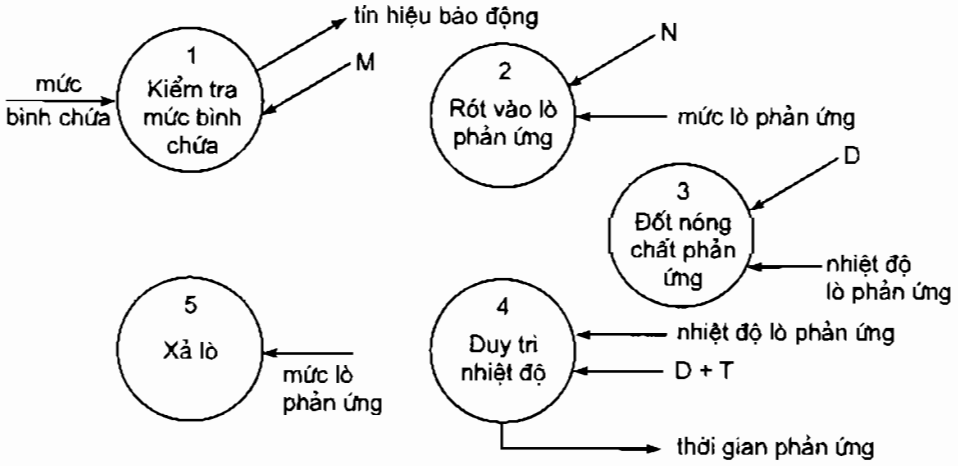


Hình VI.2 Hệ thống phản ứng nhiệt

Sự phân tích hệ thống về chức năng sẽ cho ta các BLD mức khung cảnh và mức đỉnh như trong các Hình VI.3 và VI.4. Ở đây các thiết bị vật lý (như Bình chứa, Lò phản ứng, Bộ phận đốt nóng, Van rót, Van xả, Đồng hồ) và Thao tác viên đều là đối tác của hệ thống; ngoài ra Tham số là một cơ chế cung cấp các tham số M, N, D, T cũng được xem là một đối tác. Hệ thống HĐPN chính là một phần mềm điều khiển sự hoạt động của các thiết bị trên.



Hình VI.3 BLD mức khung cảnh của HĐPN



Hình VI.4 BLD mức đỉnh của HĐPN

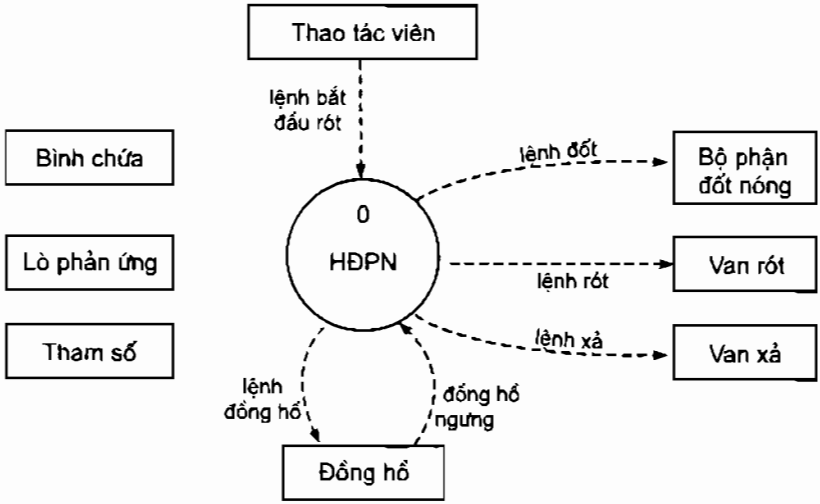
Dễ dàng nhận thấy rằng các chức năng xử lý trong BLD mức đỉnh thật là rời rạc, không gắn kết gì với nhau cả, vì thực ra chúng không trao đổi dữ liệu cho nhau. Nếu áp dụng nguyên tắc "kích hoạt bằng dữ liệu" ở đây thì ta cũng chẳng thấy một trật tự thực hiện nào của các chức năng đó.

Chất gắn kết chúng lại với nhau chính là cơ chế điều khiển chứa đựng trong các BLĐ mà ta sẽ vẽ như sau:

– *Biểu đồ luồng điều khiển khung cảnh*

Là BLĐ ở mức cao nhất, sóng đôi với BLD khung cảnh. Nó gồm 1 chức năng (diễn tả toàn HT) cùng các đối tác, và thêm các luồng điều khiển trao đổi giữa các đối tác và hệ thống.

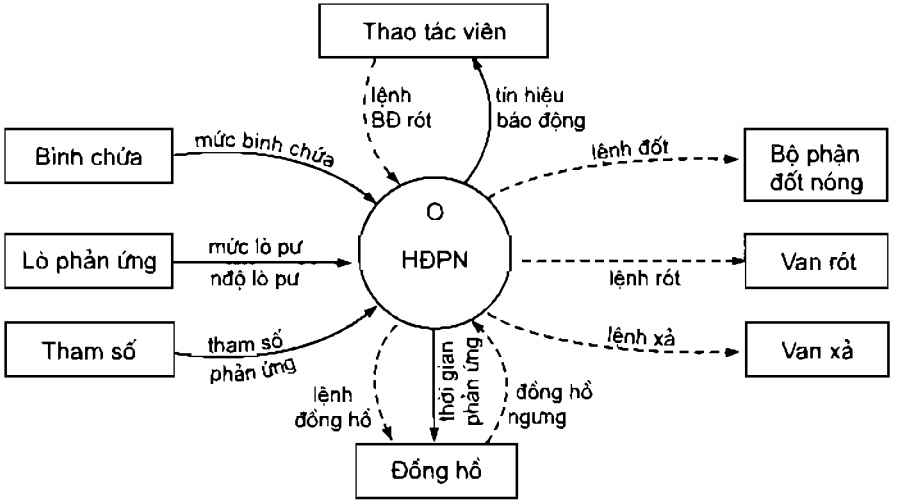
BLĐ khung cảnh của Hệ ĐK phản ứng nhiệt cho trong Hình VI.5.



Hình VI.5 BLĐ mức khung cảnh của HĐPN

Cũng có thể vẽ gộp BLĐ và BLD, vì chúng là các đồ thị có cùng các nút (đối tác, chức năng), chỉ khác nhau ở các cung (luồng dữ liệu, luồng điều khiển), song các cung đó có cách biểu diễn khác nhau, không gây nhầm lẫn.

Chẳng hạn BLD + BLĐ khung cảnh của HĐPN cho ở Hình VI.6



Hình VI.6 BLD + BLĐ mức khung cảnh của HĐPN

– Các BLĐ mức thấp

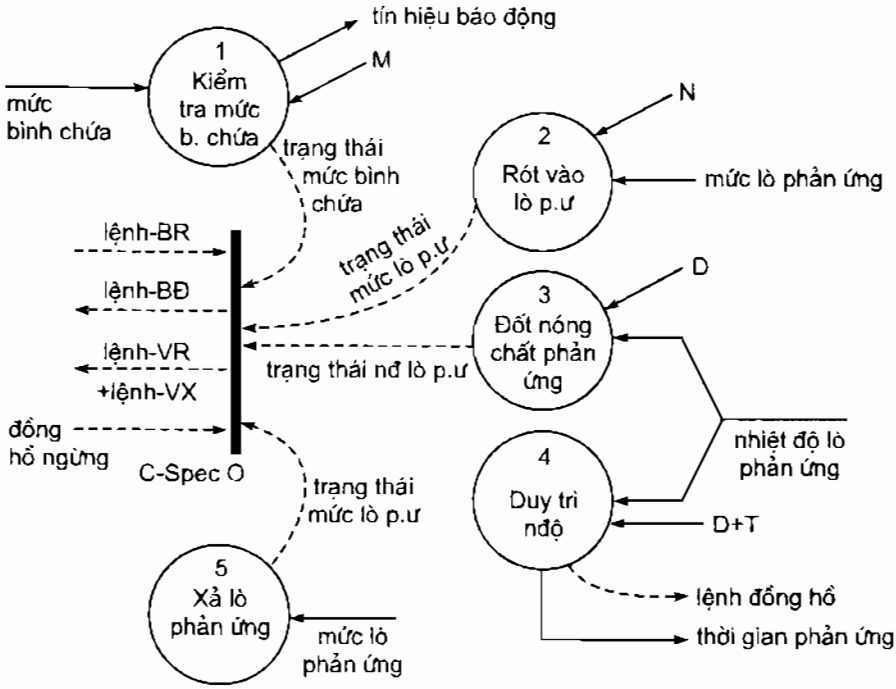
Ta cũng có 1 BLĐ mức đỉnh và dưới đó là các BLĐ mức thấp. Mỗi BLĐ này gồm có:

- cùng các chức năng xử lý như trong BLD sóng đôi với nó,
- các luồng điều khiển,
- *nhiều nhất là một* chức năng điều khiển (C–spec), biểu diễn bởi một vạch thẳng đứng, nhận và phát ra các luồng điều khiển.

Ta vẫn dùng cách đánh số các chức năng xử lý theo ký pháp chấm, và đánh số BLD và BLĐ theo số của chức năng xử lý mà chúng đã triển khai.

Thường thì người ta có thể vẽ riêng hay vẽ gộp BLĐ và BLD sóng đôi, và gọi đó là biểu đồ luồng (BL).

Thí dụ: BLD0 + BLĐ0 của HĐPN cho ở Hình VI.7.



Hình VI.7 BLDO + BLĐO của HĐPN

b) Các yếu tố hợp thành BLĐ

Đó là các luồng điều khiển, các chức năng xử lý và các chức năng điều khiển.

Các *luồng điều khiển* (cũng như luồng dữ liệu) là những tuyến truyền dẫn thông tin, có thể là thông tin nguyên thủy hay là nhóm thông tin. Các luồng điều khiển này cũng phải được bổ sung vào Từ điển với những quy ước mô tả như với các luồng dữ liệu. Chẳng hạn với HĐPN:

lệnh BR (ngth) = "Bắt đầu rót"

* lệnh bắt đầu rót của thao tác viên *

lệnh BĐ (rrạc) = [bật BĐ | tắt BĐ]

* lệnh cho bộ phận đốt nóng *

lệnh đồng hồ (ngth) = "mở đồng hồ"

* lệnh mở đồng hồ, có kèm thời gian phản ứng (luồng dữ liệu) *

lệnh VR (rrạc) = [mở VR | đóng VR]

* lệnh cho van rót *

lệnh VX (rrạc) = [mở VX | đóng VX]

* lệnh cho van xả *

trạng thái mức lò p.ư (rrạc) = ["lò p.ư rỗng"| "lò p.ư đầy"]

* trạng thái của mức lò phản ứng*

trạng thái mức b.ch. (rrạc) = ["mức b.ch. đủ" | "mức b.ch. thiếu"]

* trạng thái mức bình chứa*

trạng thái nđ lò p.ư (rrạc) = ["lò p.ư nóng"| "lò p.ư lạnh"]

* trạng thái nhiệt độ chất phản ứng
trong lò phản ứng*

tham số phản ứng = D + M + N + T

D (ngth) = * nhiệt độ được chỉ định của phản ứng*

đơn vị = độ C ; miền = 20..400;

M (ngth) = * mức tối thiểu trong bình chứa đủ cho phản ứng*

đơn vị = lít; miền = 10..5000;

N (ngth) = * mức rót chỉ dẫn vào lò phản ứng *

đơn vị = lít; miền = 10..500;

T (ngth) = * thời gian phản ứng *

đơn vị = phút; miền 15..1500;

Các *chức năng XL* chỉ chuyên biến đổi các luồng dữ liệu.

Tuy nhiên một chức năng xử lý có thể sản sinh các luồng điều khiển (từ quá trình XL các dữ liệu của nó) và đó là các điều kiện dữ liệu. Chẳng hạn, chức năng "Kiểm tra mức bình chứa" sản sinh luồng điều khiển "trạng thái mức b.ch.". P–Spec của chức năng XL này như sau:

Đầu đề

Chức năng 1: Kiểm tra mức bình chứa

Dữ liệu vào: M, mức bình chứa

Dữ liệu ra: tín hiệu báo động

Điều khiển ra: trạng thái mức b.ch.

Thân

Nếu mức bình chứa ≥ M

thì trạng thái mức b.ch. = "mức bình chứa đủ"

Không thì trạng thái mức b.ch. = "mức bình chứa thiếu"

sản sinh tín hiệu báo động

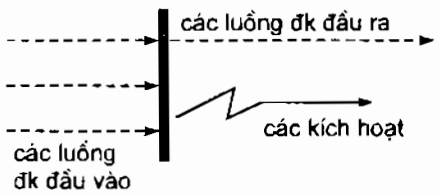
Hết Không thì

Hết Nếu

Hết Thân

Các *chức năng điều khiển* (C–Spec) chỉ chuyên xử lý các luồng điều khiển và sinh ra:

- các luồng điều khiển mới
- các kích hoạt (prompts)



Hình VI.8 Một chức năng điều khiển

Các kích hoạt biểu diễn cho sự điều khiển mà một C–Spec thực hiện đối với các chức năng XL đồng mức với nó (khởi động, chấm dứt hay đồng bộ hóa với một chức năng XL khác). Các giá trị của luồng kích hoạt có thể là (theo Ward–Mellor): Enable (Sẵn sàng), Disable (Chấm dứt), Trigger (Kích hoạt), Suspend (Tạm treo), Resume (Trở lại).

c) Sự tương hợp giữa các mức của mô hình điều khiển

– Sự phân mức các BLĐ là tương ứng với các mức BLD. Ở mỗi mức, nó sử dụng cùng các chức năng XL như BLD. Cái khác là sự có mặt các luồng điều khiển và chức năng điều khiển.

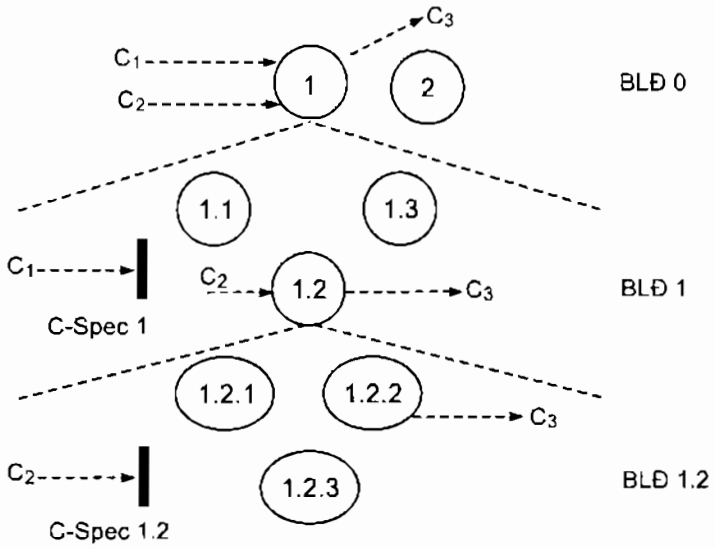
Không nhất thiết có mặt C–Spec trong một BLĐ : các luồng điều khiển có thể nhảy qua 1 mức để tác động vào C–Spec ở mức dưới.

Mặt khác một chức năng XL có thể nhận 1 luồng điều khiển ở đầu vào, luồng điều khiển này sẽ được tiêu thụ bởi C–Spec trong mức triển khai của nó, hoặc ở mức thấp hơn. Vậy 1 chức năng XL có ít nhất một luồng điều khiển ở đầu vào *nhất thiết* phải được triển khai tiếp. Mọi chức năng XL ở mức cuối cùng (không triển khai tiếp) thì không thể có luồng điều khiển ở đầu vào.

– Quy tắc về sự tương hợp giữa các mức BLĐ cũng tương tự như với BLD :

+ Các luồng điều khiển phải được bảo toàn từ mức này qua mức khác, trừ phi nó vào một C–Spec, ở đó nó đã được tiêu thụ và xử lý.

Thí dụ: Một quá trình phân rã các chức năng xử lý, cho trong Hình VI.9.



Hình VI.9 Sự bảo toàn (và tiêu thụ) các luồng điều khiển qua các mức

+ Nếu có sự phân rã các luồng điều khiển hoặc có sự đổi tên, thì các thông tin đó phải được ghi nhận vào từ điển.

§2. ĐẶC TẢ CHỨC NĂNG ĐIỀU KHIỂN (C-SPEC)

1. Nguyên lý hoạt động

Mỗi C–Spec gắn liền với một BLĐ (có thể có 1 hoặc không) và đóng vai trò "nhạc trưởng" đối với các chức năng xử lý ở cùng BLĐ đó, bởi vì chính nó quyết định khởi động hay chấm dứt mỗi chức năng xử lý.

– Quy tắc khởi động và ngưng ngắt của các chức năng xử lý (bởi các luồng kích hoạt) là như sau:

- một CNXL đã chấm dứt thì không sản sinh cái ra nữa: nó được coi như tạm thời bị loại ra khỏi hệ thống;
- một CNXL đang hoạt động, thì nó cùng mọi "tổ phụ" của nó đều đang hoạt động;
- không nhất thiết mọi CNXL đều phải nhận luồng kích hoạt: bấy giờ nó là điều khiển bởi dữ liệu (data–driven), theo sự ngầm định trong BLD của SA.

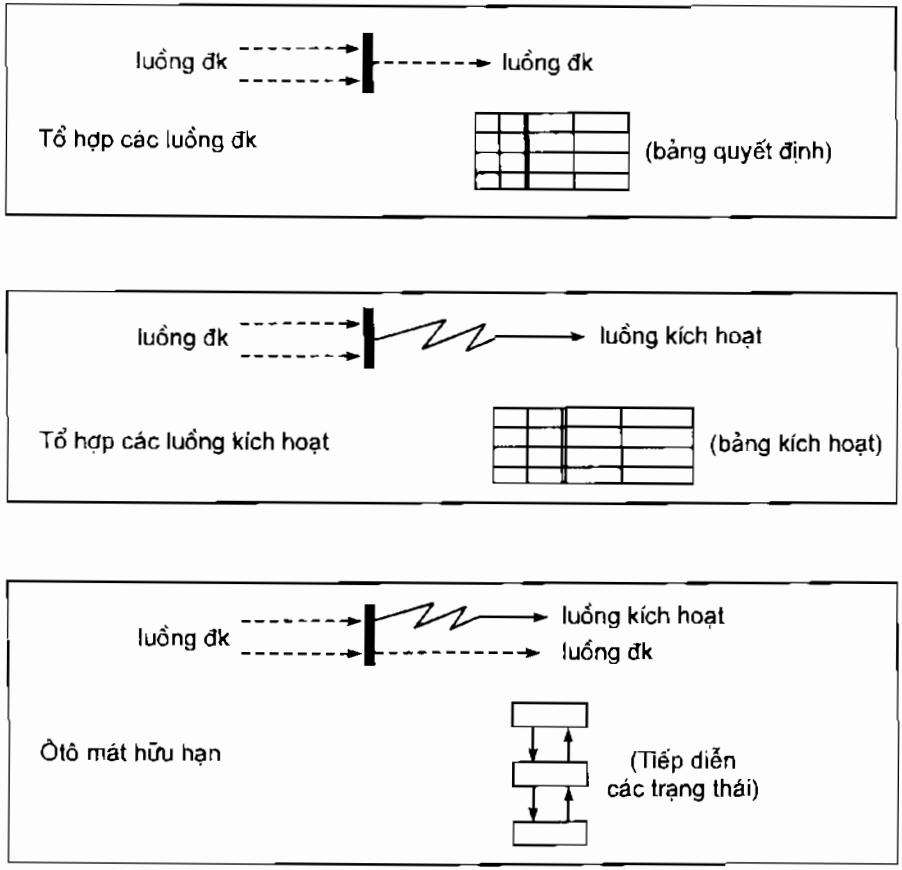
– Để thực hiện mục tiêu điều khiển của mình, các C–Spec (chỉ nhận ở đầu vào các luồng điều khiển) sẽ thực hiện các biến đổi ở các dạng sau:

- tổ hợp lôgic các luồng điều khiển đầu vào thành luồng điều khiển đầu ra (bởi một hàm Bun).
- tổ hợp lôgic các luồng điều khiển đầu vào thành các luồng kích hoạt (bởi một hàm Bun).
- sản sinh các luồng kích hoạt hay luồng điều khiển tùy thuộc sự kiện mới tới và trí nhớ (hữu hạn) về quá khứ (hành động dạng ôtômát hữu hạn).

Bởi vậy đặc tả C–Spec gồm (tất cả hay một phần):

- 1 bảng tổ hợp các luồng điều khiển;
- 1 bảng tổ hợp các luồng kích hoạt;
- 1 ôtômát hữu hạn.

Ba thành phần này (Hình VI.10) sẽ được đề cập chi tiết ở các mục tiếp sau.



Hình VI.10 Ba thành phần của C-Spec

2. Tổ hợp các luồng điều khiển

Thực hiện bởi 1 hàm Bun, biến đổi (tức thời) các luồng điều khiển đầu vào thành luồng điều khiển đầu ra. Thường được diễn tả bằng một bảng quyết định, một cây quyết định, một biểu đồ Vetch và Euler... như đã nói đối với các P–Spec.

3. Các ôtômát hữu hạn

Ôtômát hữu hạn còn gọi là "máy tiếp diễn", là máy trừu tượng cho phép biến đổi thông tin vào thành các thông tin ra, có tham khảo đến các diễn biến trong quá khứ: Các diễn biến này được ghi nhớ nhờ 1 số hữu hạn các trạng thái. Một cách hình thức, thì một ôtômát hữu hạn gồm:

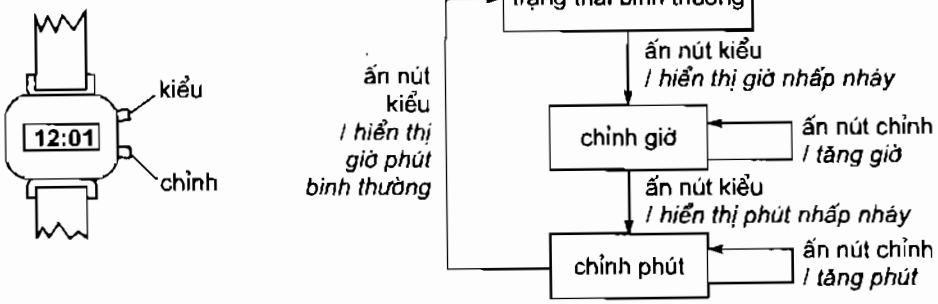
- 1 tập hữu hạn Σ các thông tin vào (còn gọi là *sự kiện*),
- 1 tập hữu hạn Q các *trạng thái*, trong đó phân biệt 1 *trạng thái đầu* q_0.

- một tập hữu hạn A các thông tin ra (còn gọi là *hành động*).
- một *hàm chuyển trạng thái* $f : Q \times \Sigma \to Q$
- một *hàm xuất* g:

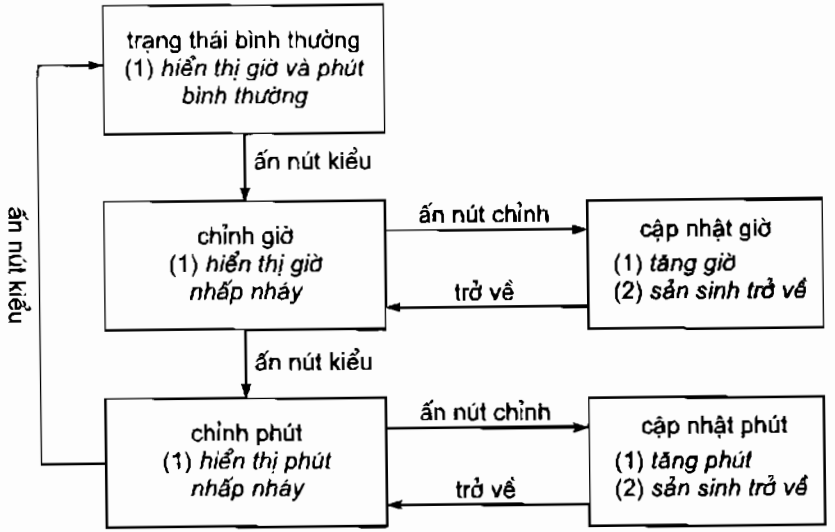
Nếu g có dạng $g : Q \times \Sigma \to A$ thì ôtômát gọi là ôtômát Mealy

Nếu g có dạng $g : Q \to A$ thì ôtômát gọi là ôtômát Moore

Thí dụ: Mô tả hành vi của một đồng hồ điện tử bằng các ôtômát hữu hạn cho dưới dạng các biểu đồ chuyển trạng thái.



Hình VI.11 Mô tả hành vi đồng hồ bằng ôtômát Mealy
(các hành động được viết xiên)



Hình VI.12 Mô tả đồng hồ bằng ôtômát Moore

Sự khác biệt giữa 2 loại ôtômát:

- Ôtômát Mealy: hành động xuất gắn với bước dịch chuyển. Hành động này bắt đầu khi sự kiện tới, và ôtômát chuyển lập tức sang trạng thái mới. Ở trạng thái mới, ôtômát lại có thể nhận 1 sự kiện và lại phát sinh một hành động mới. Vậy có hành động *song hành* nếu hành động cũ còn phải kéo dài.
- Ôtômát Moore: hành động xuất gắn với trạng thái hiện thời: Hành động này bắt đầu khi ôtômát chuyển tới trạng thái tương ứng. Ở trạng thái này, ôtômát có thể nhận sự kiện mới để chuyển sang trạng thái mới, song điều này chỉ có thể được khi hành động cũ đã hoàn tất. Vậy ở đây các hành động chỉ tiếp diễn chứ không thể song hành. Việc gắn hành động với trạng thái đã làm tăng thêm số trạng thái của ôtômát, trong đó có những trạng thái không ổn định, được đưa vào chỉ nhằm để "tải" một hành động (như hai trạng thái cập nhật giờ và phút ở ví dụ trên, ở đó cần sản sinh sự kiện nội tại "trở về").

Ngoài cách biểu diễn ôtômát bằng biểu đồ chuyển như trên, người ta còn dùng các bảng chuyển. Ví dụ đối với ôtômát Moore ở trên, ta có bảng chuyển sau:

Các trạng thái	Các sự kiện		Các hành động
	Ấn nút kiểu	Ấn nút chỉnh	
1. Trạng thái bình thường	2	không nhận	hiển thị GG : PP
2. Chỉnh giờ	3	4	hiển thị GG :
3. Chỉnh phút	1	5	hiển thị : PP
4. Cập nhập giờ	không nhận	không nhận	GG = GG+1
5. Cập nhập phút	không nhận	không nhận	PP = PP+1

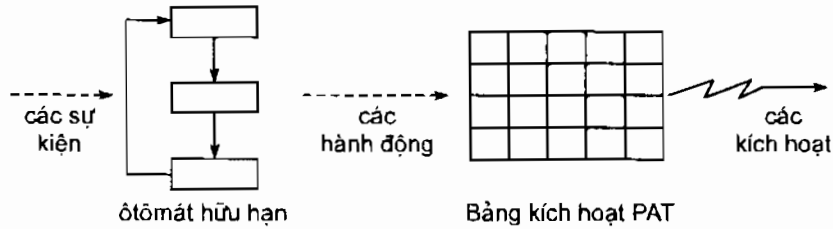
Hình VI.13 Bảng chuyển của ôtômát

4. Bảng kích hoạt các quá trình (PAT)

Các PAT (Process Activation Table) thể hiện sự điều khiển đối với các chức năng xử lý (quá trình), và chỉ rõ một quá trình là hoạt động hay không hoạt động (0/1), hoặc là sẵn sàng, chấm dứt, kích hoạt, treo, trở lại (E/D/T/S/R

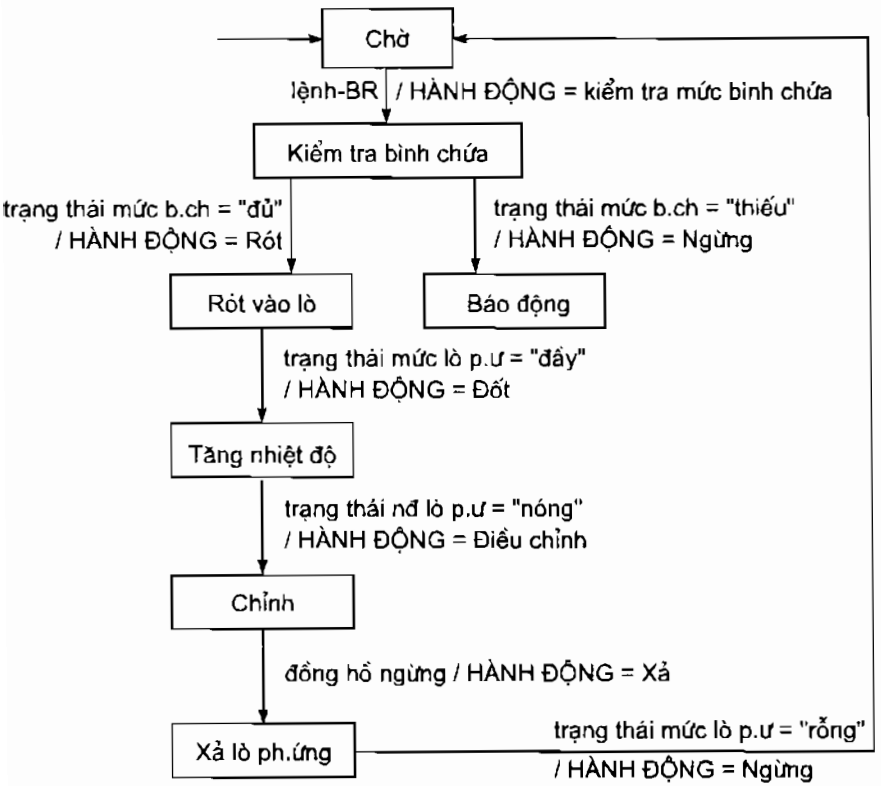
tương ứng Enable, Disable, Trigger, Suspend, Resume) hoặc là cho một trình tự khởi động nào đó (1/2/3v.v...).

Một PAT thường kết đôi với một ôtômát hữu hạn để diễn dịch các hành động xuất của ôtômát thành dạng luồng kích hoạt



Hình VI.14 Sự kết hợp giữa ôtômát và bảng kích hoạt

Thí dụ ĐK: Trở lại thí dụ của HĐPN. Trong C–Spec 0, đầu tiên ta diễn tả quá trình xử lý các luồng điều khiển bằng một ôtômát hữu hạn như trong Hình VI.15.



Hình VI.15 Ôtômát của C-Spec trong HĐPN

Các luồng điều khiển đến C–Spec này (như là: lệnh-BR, đồng hồ ngừng, trạng thái mức b.ch., trạng thái mức lò p.ư, trạng thái nđ lò p.ư) trở nên sự kiện đầu vào của ôtômát. Tuy nhiên ôtômát chỉ mới là "tầng" một của C–Spec, bởi vì các hành động xuất của ôtômát, vốn không xuất hiện trong BLĐ, chỉ là các luồng điều khiển cục bộ trong C–Spec, và phải được tiếp tục xử lý bởi bảng kích hoạt (tầng hai của C–Spec) để chuyển thành các luồng kích hoạt tới các chức năng xử lý. Các hành động này có thể xem là các giá trị của *một* luồng điều khiển gọi là HÀNH ĐỘNG

HÀNH ĐỘNG = [Kiểm tra mức bình chứa | Rót | Đốt | Điều chỉnh | Xả | Ngừng]

PAT là một bảng quyết định đặc biệt để kích hoạt các chức năng xử lý trong BLĐ, nó gồm:

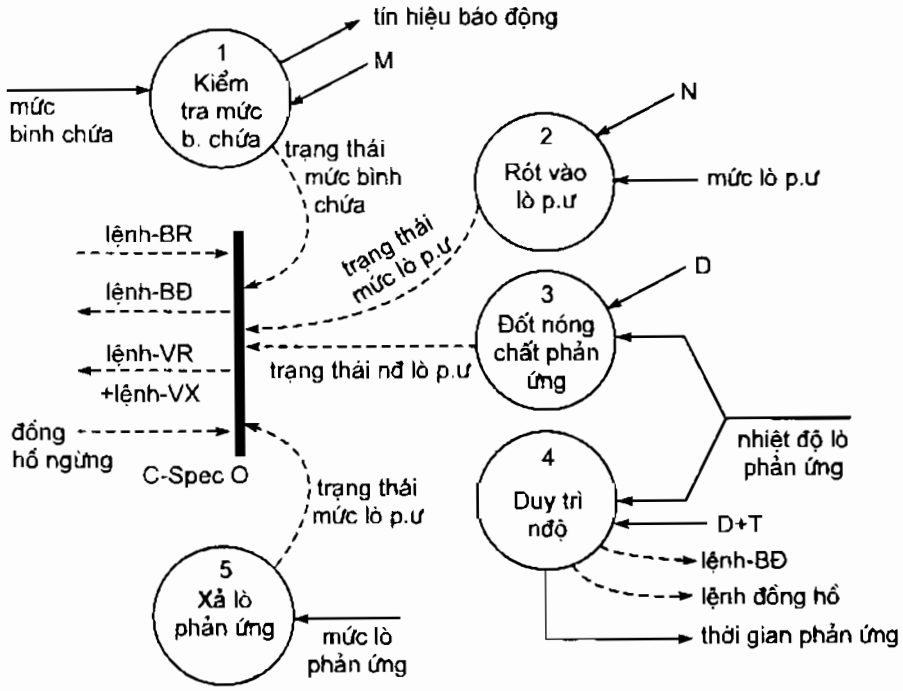
- Các cột bên phía trái chứa các cái vào: các luồng điều khiển, các sự kiện hay các hành động từ ôtômát đến, hoặc các luồng điều khiển đầu vào C–Spec và được tổ hợp cùng nhau,
- Các cột bên phía phải chứa các cái ra: các kích hoạt đến các chức năng xử lý và đôi khi còn có các luồng điều khiển liên quan.

Trong thí dụ của hệ HĐPN, đầu ra của PAT gồm các luồng kích hoạt và các luồng điều khiển xuất ra môi trường ngoài: lệnh-BĐ, lệnh-VR, lệnh-VX (dùng điều khiển bộ phận đốt và các van).

	Kích hoạt đến chức năng					Luồng ra		
HÀNH ĐỘNG	1	2	3	4	5	Lệnh - BĐ	Lệnh - VR	Lệnh - VX
Kiểm tra mức b.ch	1	0	0	0	0	"Đóng"	"Đóng"	"Đóng"
Rót	0	1	0	0	0	"Đóng"	"Mở"	"Đóng"
Đốt	0	0	1	0	0	"Mở"	"Đóng"	"Đóng"
Điều chỉnh	0	0	0	1	0		"Đóng"	"Đóng"
Xả	0	0	0	0	1	"Đóng"	"Đóng"	"Mở"
Ngừng	0	0	0	0	0	"Đóng"	"Đóng"	"Đóng"

Hình VI.16 Bảng kích hoạt trong C-Spec 0 của HĐPN

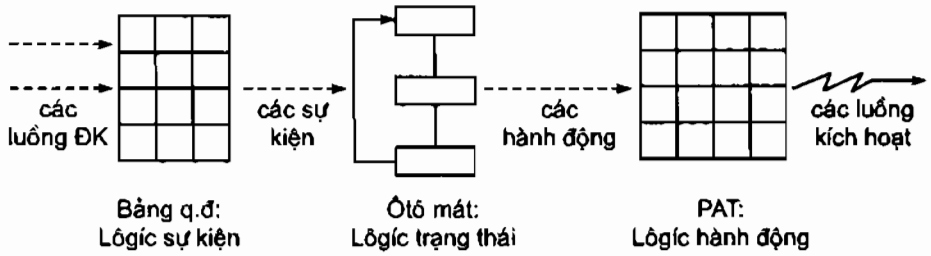
Chú ý: Khi chức năng 4 (duy trì nhiệt độ) được kích hoạt thì lệnh-BĐ là đóng hay mở chưa thể xác định được (để trắng). Việc đóng, mở đó sẽ được giải quyết ở mức thấp hơn (trong C–Spec 4), và do đó ta phải điều chỉnh lại BLĐ 0 vẽ ở mục 2 (Hình VI.7) như sau, trong đó có một luồng điều khiển ra của chức năng 4 là "lệnh-BĐ":



Hình VI.17 BLD O + BLĐ O của HĐPN với chức năng 4 đã điều chỉnh

5. Các C-Spec hỗn hợp

- Như đã nói trên, một PAT có thể được kết đôi với một ôtômát hữu hạn, tạo thành một C–Spec hỗn hợp. Ở mức hoàn chỉnh nhất thì một C–Spec có thể gồm 3 tầng:
- một tầng tổ hợp các luồng điều khiển đầu vào (lôgic sự kiện);
- một tầng ôtômát (lôgic trạng thái), tiêu thụ các sự kiện phát sinh bởi tầng trên và sản sinh các hành động;
- một tầng tổ hợp các luồng kích hoạt (lôgic hành động) biến đổi các hành động thành các luồng kích hoạt đến các chức năng xử lý ở cùng mức.



Hình VI.18 Một C-Spec hỗn hợp

§3. CÁC CHIẾN LƯỢC PHÂN TÍCH MỘT HỆ THỐNG SA-RT

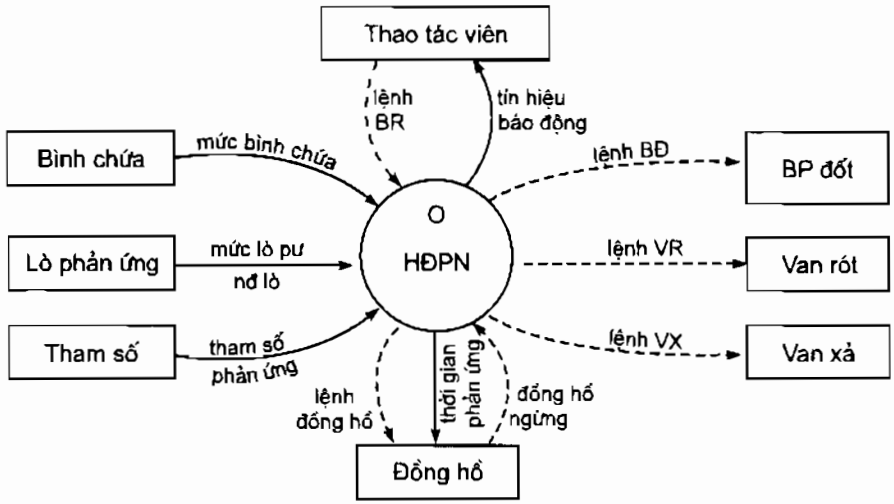
Vấn đề đặt ra là: Việc phân tích từ trên xuống cần dựa vào những cách tiếp cận nào để có được một mô hình tốt.

Kinh nghiệm của nhiều dự án SA–RT cho thấy có 7 cách tiếp cận sau:

- tiếp cận theo các chức năng,
- tiếp cận theo các luồng dữ liệu,
- tiếp cận theo các đối tượng,
- tiếp cận theo các biến cố,
- tiếp cận theo thời gian,
- tiếp cận theo các đối tác,
- tiếp cận theo kiến trúc.

Mỗi cách tiếp cận trên đều cho phép xuất phát từ biểu đồ luồng dữ liệu và điều khiển mức khung cảnh (BLD + BLĐ) thực hiện sự phân rã đầu tiên cho mức 1 (BLD 0 + BLĐ 0), và tiếp tục cho các mức dưới. Các chiến lược trên có thể dùng lẫn lộn, chẳng hạn ở mức này dùng chiến lược này, sang mức khác lại dùng chiến lược khác.

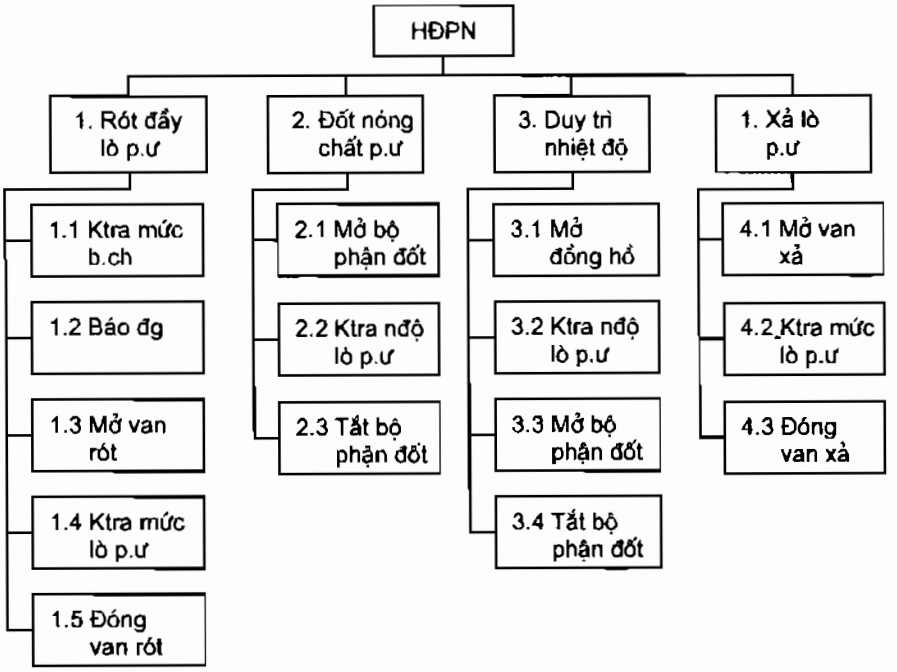
Ta sẽ tiếp tục dùng hệ HĐPN để minh họa các chiến lược phân tích khác nhau. Biểu đồ mức khung cảnh của hệ này được nhắc lại như sau:



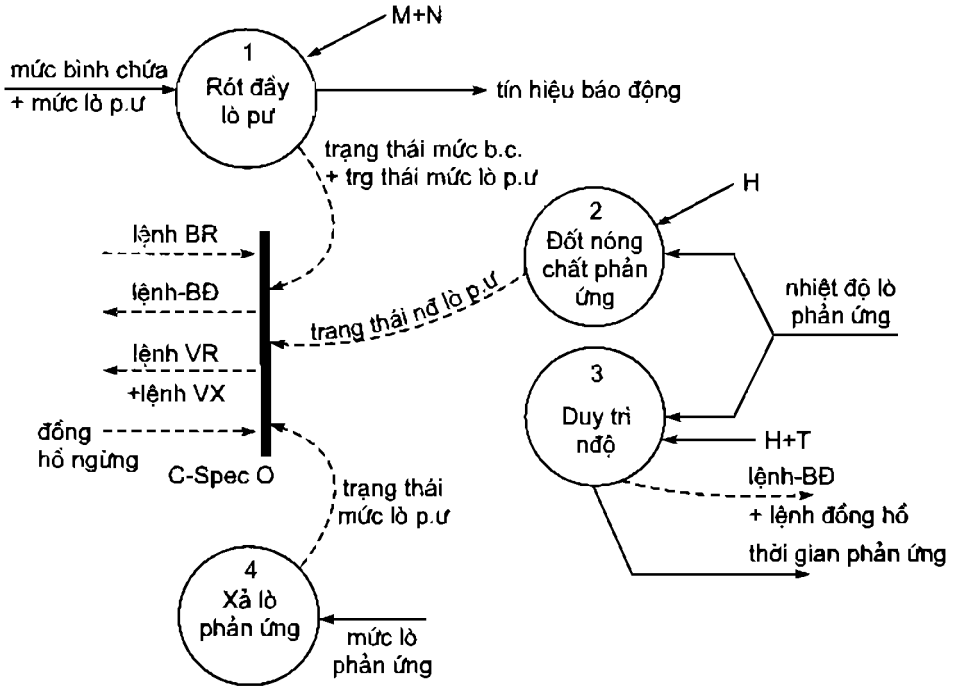
Hình VI.6 BLD +BLĐ mức khung cảnh của HĐPN

1. Cách tiếp cận theo chức năng

Cách tiếp cận này dựa trên sự nghiên cứu phân rã dần các chức năng phải thực hiện của hệ thống, theo yêu cầu của bài toán, và cho xuất hiện các chức năng chính trong biểu đồ mức đỉnh (có thể ghép lại nếu thấy cần, và giữ số lượng chức năng khoảng 7±2 cái). Sau đó bổ sung các luồng dữ liệu, luồng điều khiển và C–Spec.



Hình VI.19 Phân rã các chức năng

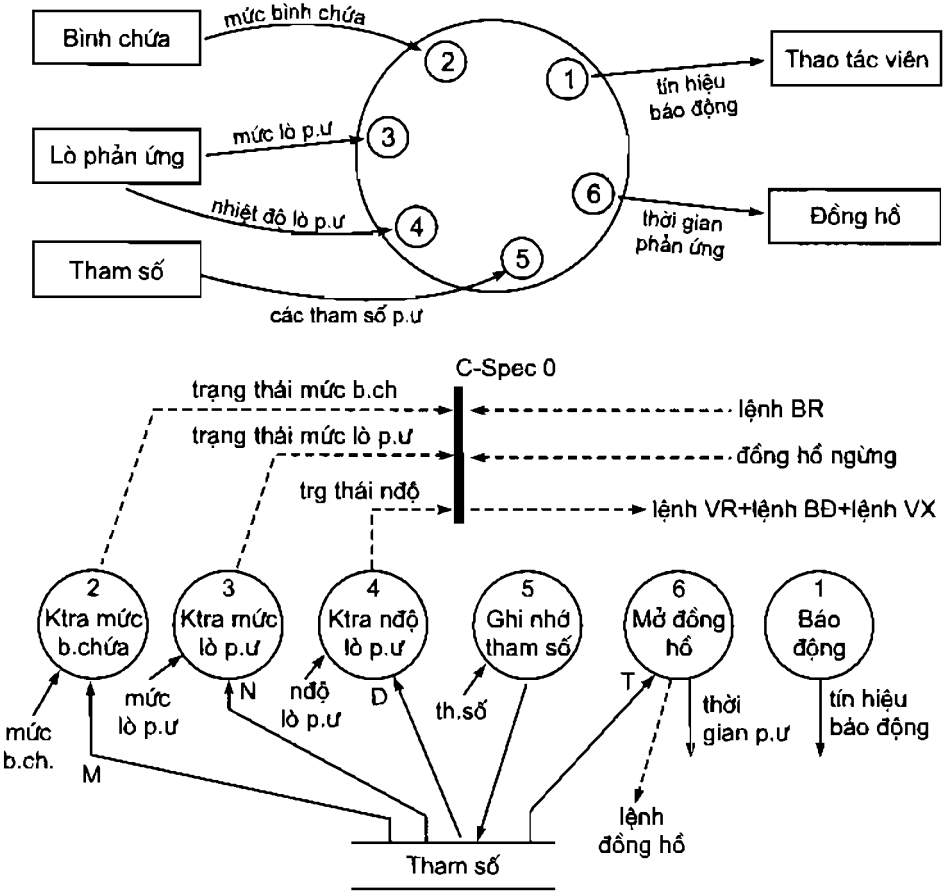


Hình VI.20 BLD 0 + BLĐ 0 của HĐPN (tiếp cận bằng phân rã chức năng)

Cách phân rã chức năng như trên dẫn tới 1 kết quả khá cân bằng. Tuy nhiên, ta thấy xuất hiện cùng 1 chức năng ở các mức khác nhau (như "kiểm tra mức lò phản ứng" và "kiểm tra nhiệt độ lò p.ư"). Điều đó là không tối ưu nhưng chấp nhận trong mô hình phân rã chức năng. Khi cài đặt sau này, có thể viết một P–Spec chung rồi tham chiếu từ các mức đến đó.

2. Cách tiếp cận theo luồng dữ liệu

Đây chính là cách tiếp cận của SA (De Macro). Khi nghiên cứu một chức năng ở một mức nào đó và tìm cách định nghĩa nó bằng một BLD, thì người ta xét các luồng dữ liệu vào và ra chức năng đó. Dõi theo các luồng đó (xuôi theo luồng vào, và ngược luồng ra) để phát hiện các chức năng xử lý chúng. Có thể gán (đầu tiên) một chức năng con xử lý cho mỗi luồng vào/ra đó, rồi đi sâu thêm vào trong (có thể ghép các chức năng con làm việc trên cùng các dữ liệu cùng bản chất làm một).

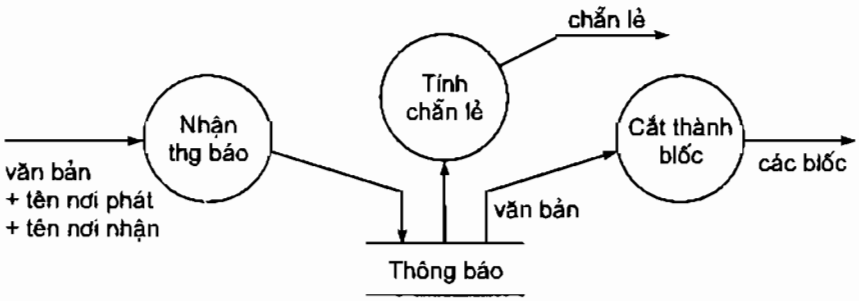


Hình VI.21 Tiếp cận theo luồng dữ liệu

Cách tiếp cận này ngay từ đầu cho 1 sự phân rã tương đối chi tiết và sau này dễ cài đặt. Song nó tương đối khó hiểu đối với người dùng. Chú ý ở đây nẩy sinh chức năng ghi nhớ tham số, không được phát hiện ở cách trên. Mọi việc điều khiển đều tập trung vào C–Spec 0, trong khi trên kia việc điều khiển được phân bố trên nhiều mức.

3. Cách tiếp cận theo đối tượng

Cách tiếp cận theo đối tượng vốn là bản chất của các phương pháp phân tích hướng đối tượng, song ngay trong SA–RT, ta cũng có thể sử dụng cách tiếp cận này để tiến hành phân tích. Đối tượng được hiểu ở đây là 1 sự gộp chung các dữ liệu với các chức năng xử lý liên quan. Chẳng hạn một đối tượng "thông báo" cho ở Hình VI.22.



Hình VI.22 Đối tượng "thông báo"

Mục đích của cách tiếp cận đối tượng trong SA–RT chủ yếu là tạo thuận lợi cho giai đoạn thiết kế, đặc biệt là với các thành phần mềm của hệ thống, khi mà ngôn ngữ cài đặt được chọn là 1 ngôn ngữ hướng đối tượng.

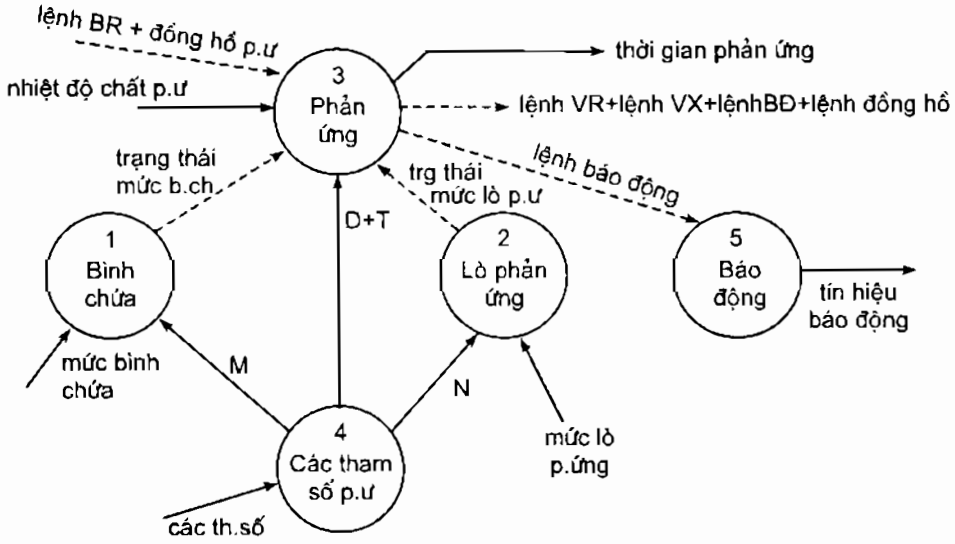
Cách tiếp cận đối tượng nhằm phát hiện các thực thể của hệ thống có cất giữ các dữ liệu chuyển tải bởi các luồng dữ liệu hay lưu trong hệ thống. Các thực thể này có thể tương ứng với các thực thể thật (như lò phản ứng, van,...) hoặc thực thể trừu tượng (phản ứng, điều khiển...). Có hai loại "đối tượng" SA–RT:

- Các đối tượng cất giữ các dữ liệu, biểu diễn bởi một chức năng trong đó có một kho dữ liệu và các chức năng con truy nhập vào kho đó;
- Các đối tượng điều khiển, biểu diễn bởi một chức năng mà bên trong có một C–Spec và có thể thêm một hay nhiều thực thể cất giữ dữ liệu.

Áp dụng cách tiếp cận đối tượng cho HĐPN, đầu tiên ta gán mỗi dữ liệu được chuyển giao bởi các luồng dữ liệu ở mức khung cảnh hay được lưu trong hệ thống với một "đối tượng" SA–RT:

Các luồng dữ liệu khung cảnh	"Đối tượng" SA–RT tương ứng
mức bình chứa	bình chứa
mức lò phản ứng	lò phản ứng
nhiệt độ chất phản ứng	phản ứng
các tham số	các tham số phản ứng
tín hiệu báo động	báo động
thời gian phản ứng	phản ứng

Ở mức đỉnh, mỗi đối tượng trên trở thành một chức năng xử lý



Hình VI.23 BL mức đỉnh của HĐPN (tiếp cận theo đối tượng)

Lưu ý rằng ở đây các chuyển giao luồng dữ liệu và điều khiển được tiết kiệm hơn: mỗi luồng "mức lò phản ứng", "lệnh-BĐ", "trạng thái mức lò p.ư" chẳng hạn chỉ được sử dụng bởi một chức năng thôi. Mặt khác nhiệm vụ điều khiển tập trung vào chức năng 3, nhưng có phối hợp cả luồng điều khiển lẫn luồng dữ liệu, cho nên được diễn tả chặt chẽ hơn C–Spec 0 ở phương pháp tiếp cận theo chức năng (ở đây ta cũng có thể san sẻ một phần điều khiển cho các chức năng khác, chẳng hạn chuyển việc điều khiển các van cho chức năng lò phản ứng (với các lệnh VR, VX)).

Ở mức dưới, các "đối tượng" được phân rã như sau:

(1) bình chứa

các dữ liệu cất giữ: mức bình chứa

các dữ liệu truy nhập : M (trên Các tham số pư)

các chức năng con:

1.1. Kiểm tra mức bình chứa

Không có C–Spec 1

(2) Lò phản ứng

các dữ liệu cất giữ: mức lò phản ứng

các dữ liệu truy nhập: N (trên Các tham số pư)

các chức năng con:

2.1. Kiểm tra mức lò phản ứng

Không có C–Spec 2

(3) Phản ứng

các dữ liệu cất giữ: nhiệt độ chất phản ứng, thời gian p.ư

các dữ liệu truy nhập: D, T (trên Các tham số pư)

các chức năng con:

3.1. Kiểm tra nhiệt độ chất phản ứng

3.2. Mở đồng hồ

có mặt C–Spec 3 (điều khiển phản ứng)

(4) Các tham số

các dữ liệu cất giữ: D, M, N, T

các dữ liệu truy nhập: Không

các chức năng con:

4.1. Ghi nhớ các tham số

Không có C–Spec 4

(5) Báo động

các dữ liệu cất giữ: tín hiệu báo động

các dữ liệu truy nhập: Không

các chức năng con

5.1. báo động

Không có C–Spec 5.

Ta gặp lại các chức năng sơ đẳng như đã có ở phương pháp luồng dữ liệu, song chúng được tổ chức một cách chặt chẽ hơn.

4. Cách tiếp cận theo biến cố

Cách tiếp cận này dựa trên sự nghiên cứu các *luồng điều khiển đi vào* một chức năng. Nó sẽ gán 1 C–Spec cho mỗi họ luồng điều khiển cùng bản chất, và mỗi C–Spec đó được đặt vào 1 chức năng xử lý (nhờ đó mà ta phát hiện các

chức năng xử lý). Phương pháp này thường dùng cho các hệ rất năng động, nghĩa là chủ yếu trao đổi nhiều biến cố với môi trường, còn việc xử lý dữ liệu là thứ yếu.

Cách tiếp cận theo biến cố phân biệt các *đường điều khiển* đồng bộ và không đồng bộ của hệ thống. Ta gọi đường điều khiển là một tập hợp các luồng điều khiển kế tiếp nhau và liên hệ với nhau một cách lôgic. Chẳng hạn, trong HĐPN thì lệnh BR của thao tác viên làm phát sinh nhiều đường điều khiển khác nhau:

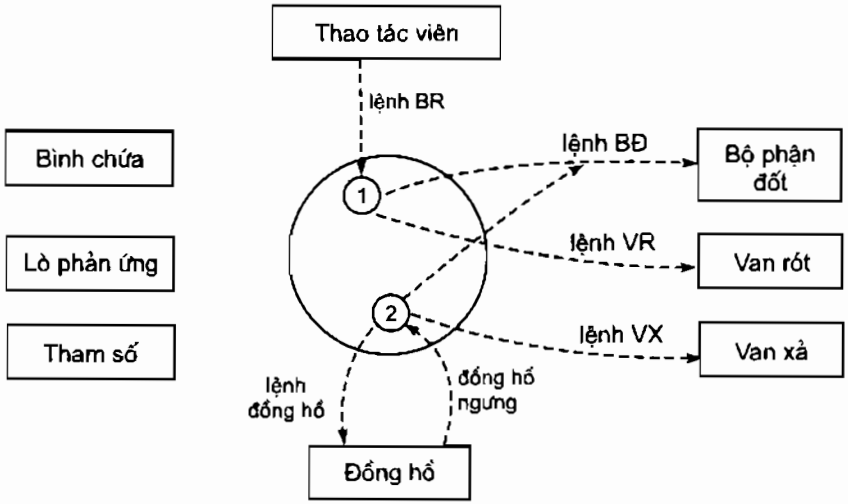
- đường 1:
  - lệnh BR
  - (– trạng thái mức b.chứa = "thiếu")
  - lệnh báo động
- đường 2:
  - lệnh BR
  - (– trạng thái mức b.chứa = "đủ")
  - lệnh VR = "MỞ"
  - (– trạng thái mức lò p.ư = "đầy")
  - lệnh VR = "ĐÓNG"
  - lệnh BĐ = "BẬT"
  - (– trạng thái nhiệt độ lò p.ư = "nóng")
  - lệnh BĐ = "TẮT", v.v...
- đường 3:
  - lệnh đồng hồ
  - đồng hồ ngừng
  - lệnh BĐ = "TẮT"
  - lệnh VX = "MỞ"
  - (– trạng thái mức lò phản ứng = "RỖNG")
  - lệnh VX = "ĐÓNG"

Các luồng điều khiển đặt trong ngoặc đơn là các luồng điều khiển nội bộ, viết thêm vào đây cho dễ hiểu, còn thực ra thì lúc mới bắt đầu phân tích thì ta chưa biết chúng và ta sẽ phát hiện ra chúng sau. Người ta phân biệt:

- Các đường điều khiển *đồng bộ*, chỉ bao gồm các biến cố được *chờ đợi* (hay được *yêu cầu*); chẳng hạn trong HĐPN thì các đường 1 và 2 là các đường điều khiển đồng bộ.
- Các đường điều khiển *không đồng bộ*, có chứa các biến cố *không yêu cầu*, hoặc là *bất ngờ* đối với hệ thống; Chẳng hạn đường 3 với luồng điều khiển "đồng bộ ngừng" là luồng điều khiển không đồng bộ.

Xuất phát từ BLĐ khung cảnh, phương pháp tiếp cận theo biến cố đầu tiên phát hiện các đường điều khiển, đồng bộ và không đồng bộ, rồi nghiên cứu xem thực sự chúng có tách biệt không hay là chúng có thể chập nhau hay tách nhánh. Sau đó cứ mỗi đường tách biệt thì lập một trung tâm điều khiển (C–Spec) và theo đó là một chức năng xử lý chứa C–Spec đó.

Áp dụng cách tiếp cận này cho HĐPN, ta phát hiện 2 đường điều khiển tách biệt, một đồng bộ (gồm đường 1 và đường 2 chập nhau) và một không đồng bộ (đường 3). Vậy ở mức đỉnh ta lập 2 chức năng xử lý tương ứng với 2 đường đó.

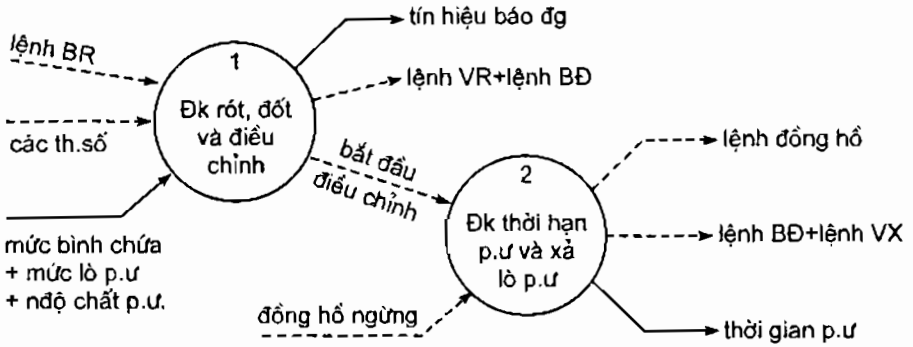


Hình VI.24 Tiếp cận theo biến cố

Sau đó, ta xác định nội dung 2 chức năng trên theo sự quản lý của chúng đối với mỗi đường điều khiển, đồng thời nối các luồng dữ liệu vào các chức

năng đó tùy thuộc mối liên hệ của chúng với các luồng điều khiển. Chẳng hạn, ở đây ta xác định nội dung 2 chức năng là:

- Chức năng 1: điều khiển việc kiểm tra mức bình chứa, việc rót vào lò phản ứng, việc đốt nóng chất phản ứng và điều chỉnh; nó khởi động chức năng 2 khi bắt đầu điều chỉnh để đặt đồng hồ.
- Chức năng 2: điều khiển đồng hồ và thời gian phản ứng, tắt bộ phận đốt nóng để kết thúc (vì lý do an toàn) và xả lò phản ứng.



Hình VI.25 BL mức đỉnh của HĐPN (tiếp cận theo biến cố).

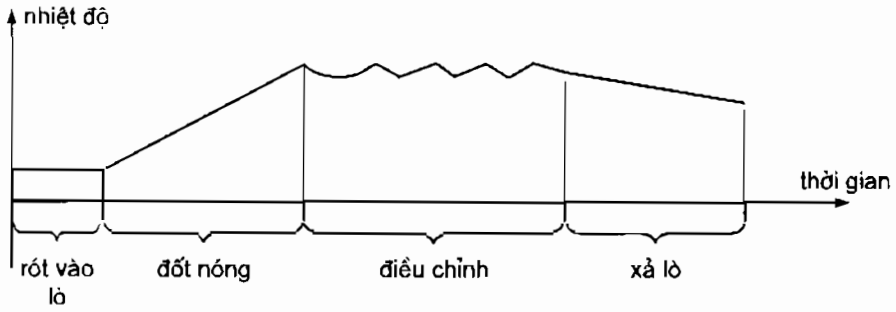
Dễ thấy rằng cách tiếp cận này cho một BLĐ khá cân bằng, bằng cách phân bố đều các luồng dữ liệu cho các chức năng, đồng thời nó cũng làm xuất hiện sớm các chức năng song song (sẽ trở thành các bộ xử lý cứng hoặc các nhiệm vụ (task) mềm sau này); chẳng hạn ở đây chức năng 2 được khởi động vào lúc bắt đầu điều chỉnh, nghĩa là lúc chức năng 1 còn làm việc tiếp.

5. Cách tiếp cận theo thời gian (hay theo sự tuần tự)

Cách tiếp cận này dựa trên sự nghiên cứu trật tự thời gian của các chức năng. Vậy nó sẽ thích hợp với các hệ thống trong đó các chức năng được khởi động và thực hiện theo một trật tự nhất định, và không thích hợp với các hệ thống thực sự thời gian thực, ràng buộc mạnh vào các biến cố bất chợt và trật tự thực hiện các chức năng là không cố định.

Tuy nhiên trong nhiều trường hợp, việc phát hiện các chức năng dần dần theo trật tự thời gian là rất tự nhiên và phương pháp này là rất thuận lợi.

Với HĐPN, ta có một thời biểu như trong Hình VI.26, cho thấy các chức năng thực hiện trong mỗi giai đoạn



Hình VI.26 Thời biểu của HĐPN

Sau đây là một thí dụ khác mà thoạt đầu ta chưa thấy rõ các chức năng của hệ thống.

Tủ bán đồ uống tự động: Để tránh tối đa mọi sự gian lận và phá hoại, tủ tự động phải có các khả năng sau:

– Nhận các "vật" do khách đưa vào để trả tiền nước uống.

– Kiểm tra kích cỡ, độ dầy và cạnh của các "vật" để nhận diện đồng tiền thật.

– Chỉ lấy các đồng 5, 2 và 1 hào, trả lại các đồng khác.

– Chỉ chấp nhận trả dần tiền và chọn thức uống sau khi phát hiện được một đồng đúng đầu tiên.

– Nhận sự lựa chọn thức uống của khách.

– Kiểm tra thức uống do khách lựa chọn là có sẵn hay không (tủ có thể rỗng). Nếu không có thì báo cho khách và trả lại tiền.

– Tủ có nhiều thức uống. Các thức uống này và giá của chúng có thể thay đổi khi bảo dưỡng tủ.

– Khi khách phủ định sự lựa chọn của mình và thôi không chọn tiếp, thì trả lại tiền cho khách.

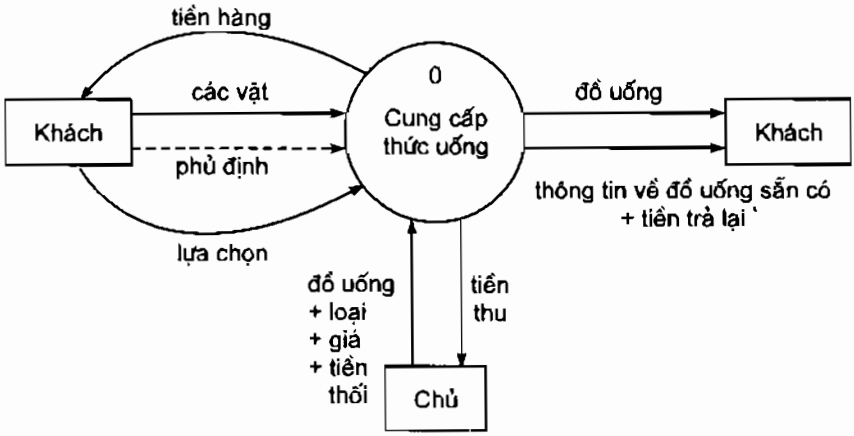
– Cung cấp thức uống nếu có sẵn và tiền trả đã đủ.

– Trả lại tiền thừa nếu tổng số tiền cho vào cao hơn giá thức uống.

– Không chấp nhận sự lựa chọn mới sau khi thức uống đã được cung cấp, trừ phi khách lại đưa thêm một đồng tiền đúng.

– Có sẵn các đồng tiền trong két để có thể thối lại tiền.

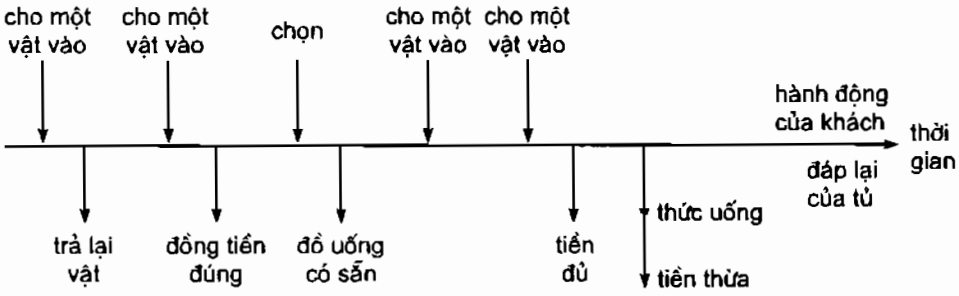
Biểu đồ khung cảnh của tủ tự động này là khá đơn giản (Hình VI.27)



Hình VI.27 BL khung cảnh

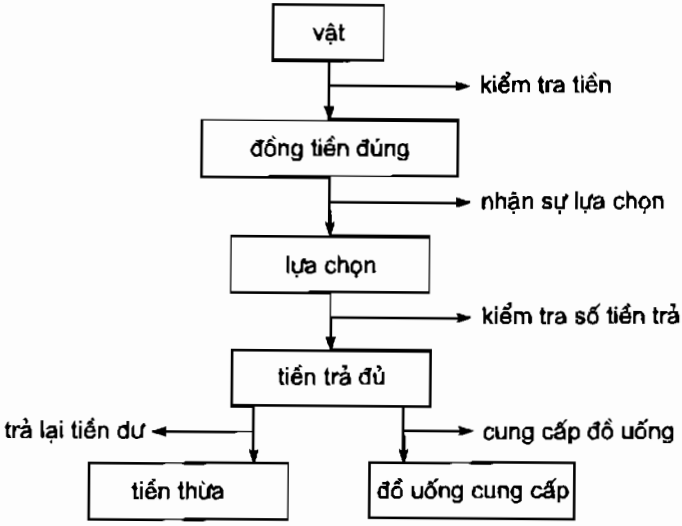
Tuy nhiên các chức năng của hệ thống thì không dễ phát hiện nếu không có cách tiếp cận thích hợp.

Với cách tiếp cận theo thời gian đầu tiên ta vẽ một thời biểu cho tủ tự động như trong Hình VI.28.



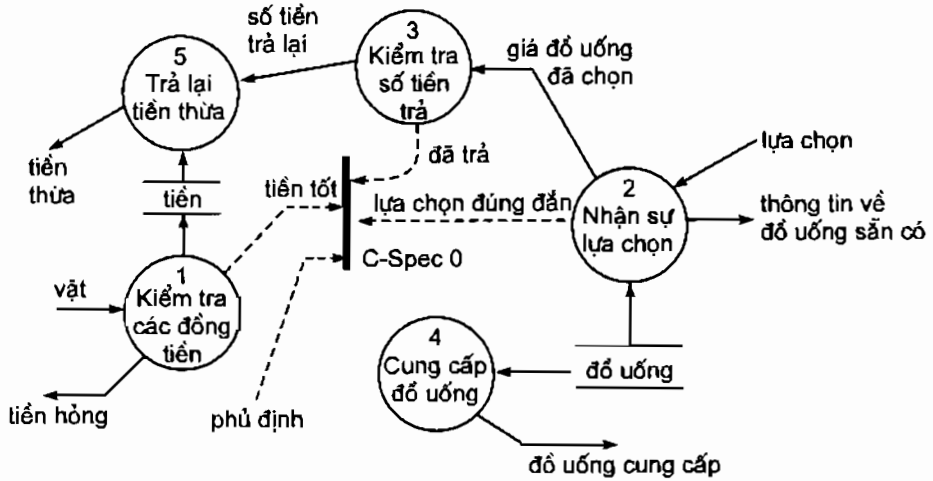
Hình VI.28 Thời biểu của hệ cung cấp thức uống

Từ thời biểu này (còn gọi là kịch bản sử dụng của hệ thống), ta vẽ một biểu đồ *phụ thuộc chức năng:* mỗi nút là một dữ liệu, cung nối hai nút diễn tả sự phụ thuộc về chức năng giữa chúng (một chức năng đã được thực hiện). Biểu đồ này là một thứ tự bộ phận, chứ không là một thứ tự toàn phần, vì có những nhánh triển khai song song.



Hình VI.29 Biểu đồ phụ thuộc chức năng

Biểu đồ trên cho phép phát hiện các chức năng chính của hệ thống, và vẽ biểu đồ luồng dữ liệu mức đỉnh của hệ thống như ở Hình VI.30.



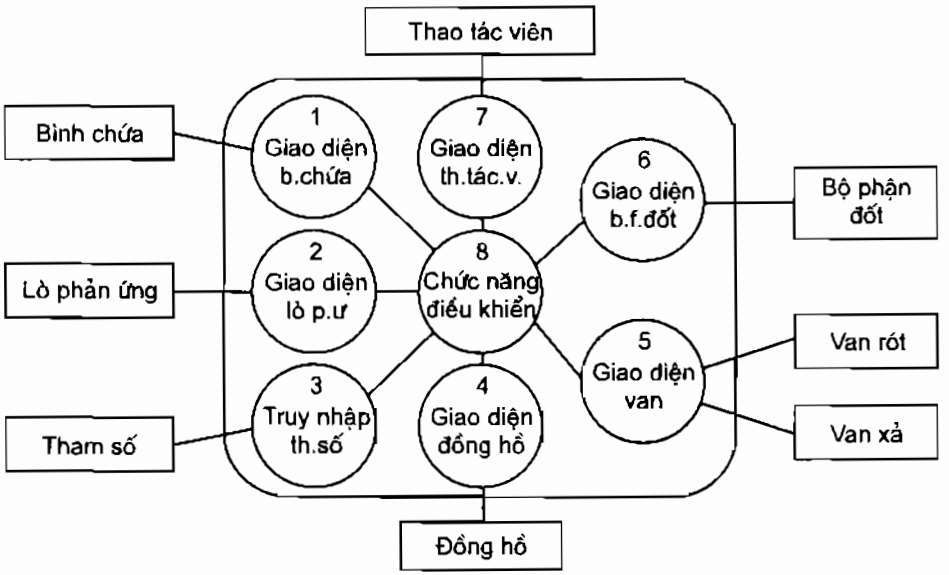
Hình VI.30 Biểu đồ luồng mức đỉnh của hệ cung cấp thức uống
(tiếp cận theo thời gian)

6. Cách tiếp cận theo các đối tác

Cách tiếp cận này dựa trên các đối tác được diễn tả trong biểu đồ khung cảnh. Cứ mỗi đối tác thì lập một chức năng giao diện. Các giao diện với các đối tác cùng loại có thể sát nhập lại. Ngoài ra thêm một chức năng trung tâm giữ

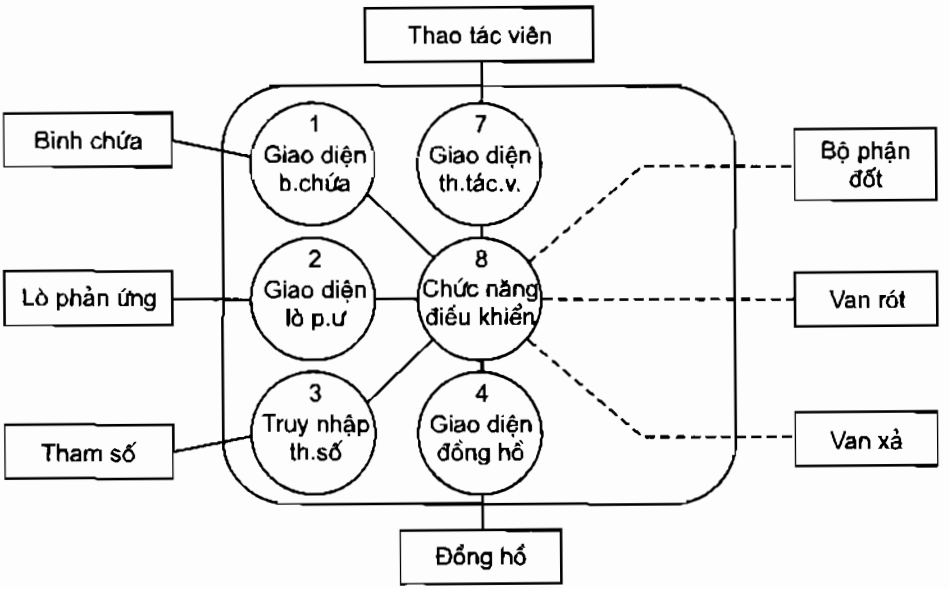
vai trò điều khiển chung. Các chức năng giao diện không những thực hiện sự đối thoại với các đối tác, mà còn thực hiện mức xử lý sơ bộ các dữ liệu trao đổi.

Với HĐPN, sự chia cắt theo đối tác cho ta BLD ở Hình VI.31



Hình VI.31 Tiếp cận theo đối tác

Tiếp đó ta có thể tinh giản sơ đồ bằng cách loại bớt các chức năng giao diện chỉ trao đổi các luồng điều khiển với các đối tác.



Hình VI.32 Sơ đồ đã tinh giản

Vậy ở mức đỉnh, ta đã làm xuất hiện các chức năng như sau:

(1) Giao diện bình chứa

dữ liệu xử lý : mức bình chứa

dữ liệu truy nhập: M (trên chức năng 3, Truy nhập tham số)

điều khiển phát sinh: trạng thái mức bình chứa (nội bộ)

chức năng con :

1.1. Kiểm tra mức bình chứa

(2) Giao diện lò phản ứng

dữ liệu xử lý : mức lò, nhiệt độ chất phản ứng

dữ liệu truy nhập: N, D (trên 3, Truy nhập tham số)

chức năng con :

2.1. Kiểm tra mức lò phản ứng

2.2. Kiểm tra nhiệt độ chất phản ứng

(3) Truy nhập các tham số

dữ liệu lưu giữ : M, N, D, T

dữ liệu truy nhập: không

chức năng con :

3.1. Lưu giữ các tham số

(4) Giao diện đồng hồ

dữ liệu xử lý : thời gian phản ứng

dữ liệu truy nhập: T (trên 3, Truy nhập tham số)

điều khiển phát sinh: lệnh đồng hồ (ngoài)

điều khiển thu nhận: đồng hồ ngừng (ngoài)

chức năng con :

(4.1) Khởi động đồng hồ

(7) Giao diện thao tác viên

dữ liệu xử lý : tín hiệu báo động

dữ liệu truy nhập: không

điều khiển thu nhận: lệnh bắt đầu rót (ngoài)

chức năng con :

(7.1) Phát báo động

(8) Chức năng điều khiển

dữ liệu xử lý : không

dữ liệu truy nhập: không

điều khiển thu nhận: mọi điều khiển nội bộ

điều khiển phát sinh: lệnh BĐ, lệnh VR, lệnh VX (ngoài)

chức năng con : không

chức năng này thực chất là C–Spec 0 (điều khiển phản ứng).

Danh sách này gần giống với danh sách thu được từ cách tiếp cận đối tượng và lại cho cùng các chức năng sơ đẳng như mọi cách tiếp cận khác. Tuy nhiên ở đây các chức năng chỉ chịu trách nhiệm trao đổi thông tin với các đối tác, cho nên chúng đã không được tổ chức chặt chẽ (thể hiện sự gắn bó giữa dữ liệu và chức năng) như trong cách tiếp cận đối tượng.

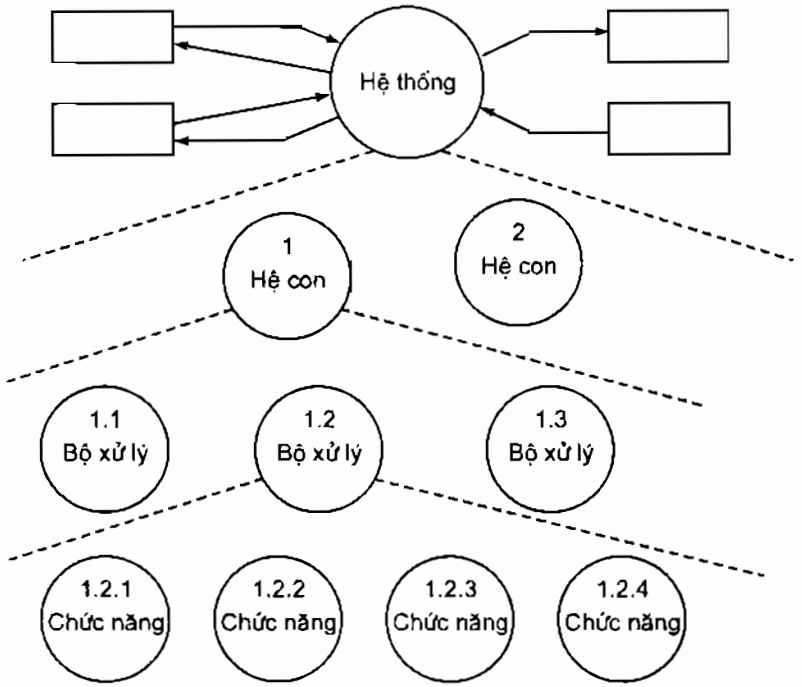
7. Cách tiếp cận theo kiến trúc

Cách tiếp cận này dựa trên các ràng buộc có sẵn về thiết kế kiến trúc của hệ thống. Vậy nó đối lập với cách tiếp cận chức năng, ở đó hệ thống được xem xét một cách trừu tượng, thoát ly khỏi mọi sự cài đặt vật lý. Việc đề cập sớm, ngay trong phân tích, các ràng cuộc về kiến trúc đưa đến hai hệ quả đối lập:

– hoặc sẽ cho một sự chia cắt xác đáng, dễ cài đặt và tiết kiệm được thời gian khi thiết kế;

– hoặc ngược lại, cho một sự chia cắt không hợp lý, và đến khi cài đặt thì mới phát hiện thấy nhiều thiếu sót, phải phá vỡ để làm lại.

Tuy nhiên cách tiếp cận này là thích hợp trong trường hợp mà kiến trúc của hệ thống đã được xác định từ đầu, không bàn cãi nữa. Ở đây sự chia cắt ở mức 0 không phải thành các chức năng, mà thành các hệ thống con, hay trực tiếp thành các bộ xử lý vật lý của hệ thống. Rồi đến các mức dưới mới chia cắt thành các quá trình xử lý, tức là các chức năng, phân phối cho các bộ xử lý nói trên.



Hình VI.33 Tiếp cận theo kiến trúc

Chú thích: Các cách tiếp cận khác nhau kể trên đều nhằm mục đích phân chia hệ thống ở những mức phân cấp đầu tiên, đặc biệt là mức 0 và mức 1. Tuy nhiên ở các mức thấp ta vẫn có thể áp dụng các chiến lược chia cắt đó, thậm chí áp dụng chiến lược này cho mức này và chiến lược khác cho mức khác. Nhưng rốt cục, như trong thí dụ HĐPN, ta phải thu được một tập hợp các chức năng sơ đẳng như nhau (các P–Spec). Vậy sự phân chia trên xuống theo các cách tiếp cận khác nhau, xét từ dưới lên, chỉ là các sự gom nhóm khác nhau của cùng một tập hợp các chức năng sơ đẳng.

Chương VII
THIẾT KẾ HỆ THỐNG

Thiết kế là nhằm chuyển các đặc tả lôgic của hệ thống (về chức năng, về dữ liệu và về động thái) thành các đặc tả vật lý của hệ thống, có tính tới các yêu cầu và ràng buộc vật lý. Như vậy, nếu phân tích nhằm trả lời câu hỏi "Là gì?", thì thiết kế nhằm trả lời câu hỏi: "Như thế nào?".

Đầu vào của công việc thiết kế bao gồm:

- Các đặc tả lôgic về hệ thống, có được từ giai đoạn phân tích;
- Các yêu cầu và ràng buộc về các điều kiện vật lý cụ thể, như là các hình trạng phần cứng, phần mềm, các tài nguyên, các dung lượng có thể, các yêu cầu về thời gian thực hiện, thời gian trả lời, về xử lý sai lỗi, về chi phí bảo trì v.v...

Đầu ra của công việc thiết kế sẽ là các quyết định về:

- Một kiến trúc tổng thể của hệ thống;
- Các hình thức trao đổi trên biên của hệ thống (các mẫu thu thập, các tài liệu in ra, các giao diện người/máy).
- Các kiểm soát, nhằm phòng ngừa các sự cố vật lý và các ý đồ phá hoại;
- Tổ chức vật lý của cơ sở dữ liệu theo các phương án sử dụng tệp hay sử dụng hệ quản trị cơ sở dữ liệu có sẵn;
- Tổ chức chương trình theo các môđun.

Các quyết định về thiết kế luôn luôn là một sự thỏa hiệp giữa hai mặt: tính hợp lý đơn thuần (lôgic) và sự hạn chế của điều kiện cụ thể (vật lý).

Sau đây ta sẽ lần lượt đi sâu vào các công việc khác nhau trong thiết kế. Thứ tự trình bày không nhất thiết là thứ tự thực hiện, vì thực ra các công việc này thường đan xen, níu kéo nhau, không thể thực hiện một cách tuyến tính được. Nội dung trình bày là dựa theo phương pháp SD (structured design) do E. Yourdon và L. Constantine đề xuất.

§1. THIẾT KẾ TỔNG THỂ

1. Mục đích

Mục đích của thiết kế tổng thể là nhằm đưa ra một kiến trúc tổng thể của hệ thống. Kiến trúc này thể hiện sự phân chia hệ thống thành nhiều hệ thống con và sự chia tách phần thực hiện bằng thủ công với phần thực hiện bằng máy tính (một máy tính hay nhiều máy tính) trong mỗi hệ thống con đó.

2. Phân chia hệ thống thành các hệ thống con

Hệ thống con (cũng có khi gọi là gói) là một sự gom nhóm các chức năng (hay các chương trình) trong một hệ thống xung quanh một nhiệm vụ hay một mục đích nào đó.

Sự phân chia hệ thống thành các hệ thống con là nhằm giảm thiểu sự phức tạp, sự cồng kềnh, hoặc nhằm tạo ra những thuận lợi cho quá trình thiết kế cũng như khai thác, bảo dưỡng sau này.

Sự phân chia hệ thống thành các hệ thống con được tiến hành ngay trên biểu đồ luồng dữ liệu của hệ thống lập từ giai đoạn phân tích. Ta dùng một đường ranh (một đường đứt nét chẳng hạn) để tách các chức năng trong biểu đồ luồng dữ liệu thành nhóm, mỗi nhóm là một hệ thống con. Thông thường thì các BLD cấp cao (mức đỉnh hay mức dưới đỉnh) cho ta những gợi ý tốt: mỗi chức năng lớn xuất hiện trong các BLD mức cao đó là đại diện cho một hệ thống con, gồm các chức năng phân rã từ nó trong các BLD mức thấp. Tuy nhiên sự phân chia dựa theo chức năng đó phải được xem xét dựa theo hai tiêu chuẩn:

- Tính cố kết cao: Cố kết (cohesion) là sự gắn bó về lôgic hay về mục đích của các chức năng trong cùng một hệ thống con. Sự cố kết của các chức năng trong cùng một hệ thống con phải càng cao càng tốt.
- Tính tương liên yếu: Tương liên (coupling) là sự trao đổi thông tin và tác động lẫn nhau giữa các hệ thống con. Một sự phân chia tốt đòi hỏi sự tương liên lẫn nhau giữa các hệ thống con phải càng lỏng lẻo, càng đơn giản càng tốt.

Ta sẽ có dịp bàn sâu hơn về hai tiêu chuẩn này khi đề cập đến các môđun chương trình trong §5.

Sự phân chia hệ thống thành hệ thống con thực ra không phải dựa hoàn toàn vào căn cứ chức năng thuần túy, mà còn có nhiều căn cứ khác phải tham khảo đến, đặc biệt là các căn cứ từ thực tế. Bởi vậy sẽ có nhiều cách gom nhóm, mà ta có thể kể ra một số cách như sau:

(1) Gom theo thực thể: Gom tụ vào một hệ thống con những chức năng liên quan tới cùng một (hay một số) kiểu thực thể nhất định.

Ví dụ: Hệ con KHÁCH HÀNG bao gồm mọi chức năng liên quan tới khách hàng, như các chức năng về xử lý đơn đặt hàng, làm hóa đơn, phát hàng, thanh toán, xử lý nợ đọng v.v... Hệ con KHO VẬT TƯ quy tụ các chức năng xuất, nhập hàng, đặt hàng bổ sung, kiểm kê v.v...

(2) Gom theo sự kiện giao dịch: Gom tụ vào một hệ thống con những chức năng được khởi động (trong thời gian) khi có một sự kiện giao dịch nào đó xảy ra.

Ví dụ: Khi một đơn đặt hàng đến (một sự kiện giao dịch) thì một loạt các chức năng sẽ được khởi động, như Ghi nhận đơn hàng, Kiểm tra khả năng đáp ứng của kho hàng, Kiểm tra tư cách khách hàng, Xử lý các yêu cầu của đơn hàng, v.v... Các chức năng này được gom vào một hệ thống con XỬ LÝ ĐƠN HÀNG.

(3) Gom theo trung tâm biến đổi: Nếu phát hiện trong BLD có một nhóm các chức năng cộng tác với nhau để thực hiện một tính toán hay một sự biến đổi thông tin đặc biệt nào đó, thì tách chúng ra thành một hệ thống con.

Ví dụ: Hệ tính lương, hệ làm báo cáo hằng tháng...

(4) Gom theo một lý do thiết thực nào đó:

Đó có thể là những lý do nhằm:

- thuận lợi cho cấu trúc kinh doanh;
- thuận lợi cho vị trí của công sở;
- thích ứng với cấu hình phần cứng vốn có;
- phù hợp với trình độ đội ngũ cán bộ;
- phù hợp với sự phân cấp trách nhiệm công tác;
- tạo ra khả năng bảo mật tốt hơn v.v...

Thí dụ QL: Hệ cung ứng vật tư (tiếp theo)

Xuất phát từ BLD mức đỉnh của hệ CƯVT đã lập được ở giai đoạn phân tích (Hình III.34) ta phân hệ thống này thành các hệ thống con. Căn cứ của sự phân chia ở đây là cấu hình phần cứng vốn có. Bởi muốn giữ lại hai máy tính

cũ (cùng với phần lớn chương trình có sẵn trên đó), mà ngay từ giai đoạn Tìm hiểu nhu cầu (Chương II), ta đã chấp nhận giải pháp 5 (trang 55) trên đó ta giữ lại hai hệ con là hệ ĐH và hệ PH có điều chỉnh ít nhiều so với tình trạng cũ, mỗi hệ sẽ vẫn tọa lạc trên một máy tính.

Giờ đây, trên BLD ở Hình III.34, ta vạch ranh giới cho hai hệ con đó. Riêng chức năng 4 (Khớp đơn hàng với hàng về) đặt ở hệ thứ nhất, nhưng tách một phần việc là Tìm địa chỉ phát hàng đưa sang hệ thứ hai (phù hợp với ý đồ điều chỉnh của Giải pháp 5), trở thành chức năng 4'.

Đặt lại tên để phản ánh rõ nội dung hơn, ta có:

- HT1: Đặt hàng và theo dõi đơn hàng, bao gồm bốn chức năng 1, 2, 4 và 6.
- HT2: Quản lý kho, nhận và phát hàng, bao gồm năm chức năng 8, 9, 3, 4' và 5.

Chức năng 7 (Thanh toán) chuyển sang phòng Tài vụ và không còn nằm trong hệ thống CƯVT nữa.

Lại nhận thấy mỗi hệ con HT1, HT2 trên bao gồm hai nhiệm vụ tách biệt, vậy lại có thể chia mỗi hệ đó làm đôi, kết quả là có bốn hệ thống con như sau:

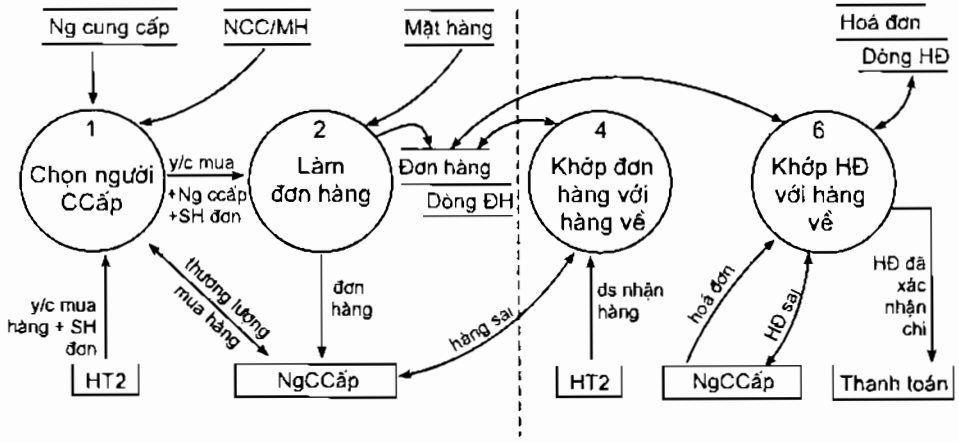
+ HT1.1 : Đặt hàng

+ HT1.2 : Theo dõi đơn hàng

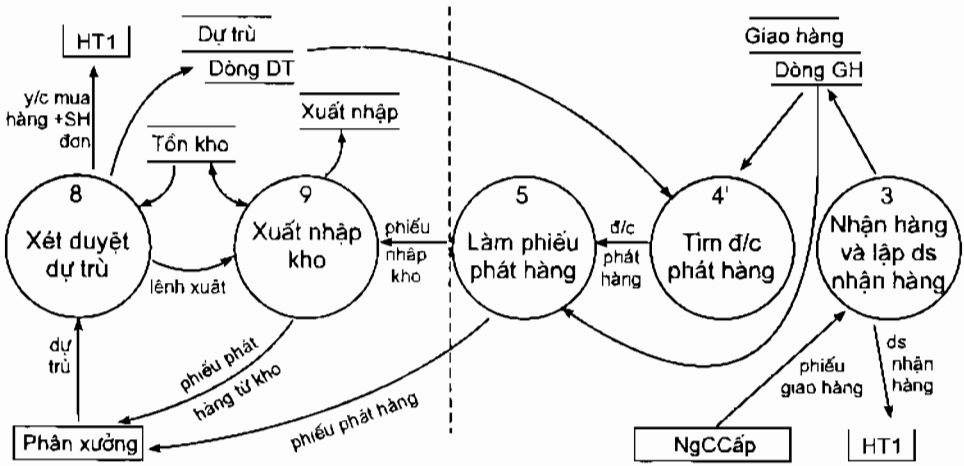
+ HT2.1 : Quản lý kho dự trữ

+ HT2.2 : Nhận và phát hàng

Sự chia tách trên được trình bày trong hai hình VII.1 và VII.2.



Hình VII.1 Các hệ con 1.1 và 1.2



Hình VII.2 Các hệ con 2.1 và 2.2

3. Phân định phần thực hiện thủ công với phần thực hiện bằng máy tính

Cho đến nay, qua bước phân tích, ta đã mô tả hệ thống với nhiều chức năng, nhiều kho dữ liệu, nhưng ta chưa hề bao giờ để cập đến các câu hỏi như chức năng nào sẽ do máy tính thực hiện, chức năng nào vẫn do con người thực hiện, kho dữ liệu nào được lưu trong máy tính (dưới dạng tệp hoặc cơ sở dữ liệu) và kho dữ liệu nào vẫn được quản lý bằng tay (dưới dạng hồ sơ, sổ sách).

Giờ đây đã đến lúc ta phải quyết định rõ về vai trò của máy tính trong hệ thống, bằng cách phân định trên biểu đồ luồng dữ liệu phần thực hiện bằng máy tính và phần thực hiện thủ công; đồng thời cũng quyết định phương thức xử lý theo mẻ được vận dụng ở những nơi nào và phương thức xử lý trực tuyến được vận dụng ở những nơi nào.

Sự phân định đó cũng được thực hiện bằng cách vạch một đường ranh (một đường đứt nét chẳng hạn) để chia tách phần máy tính với phần thủ công trên BLD. Cách làm là như sau:

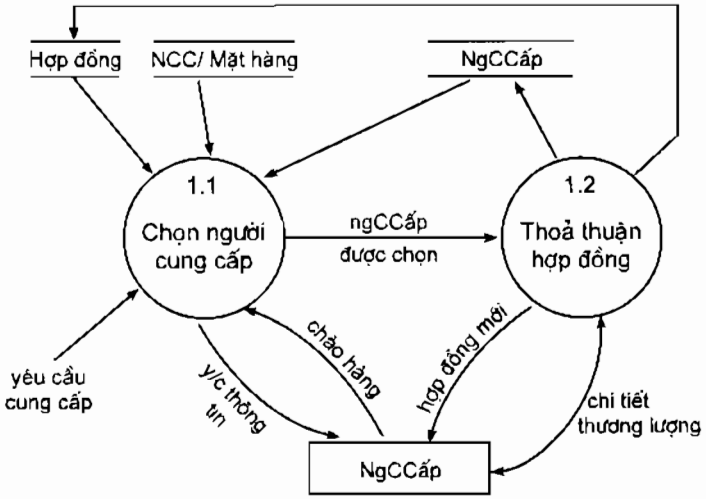
(i) Đối với các chức năng:

Xem xét từng chức năng trong BLD để quyết định chức năng nào sẽ thực hiện bằng máy tính, chức năng nào sẽ thực hiện bởi con người. Hai trường hợp xảy ra:

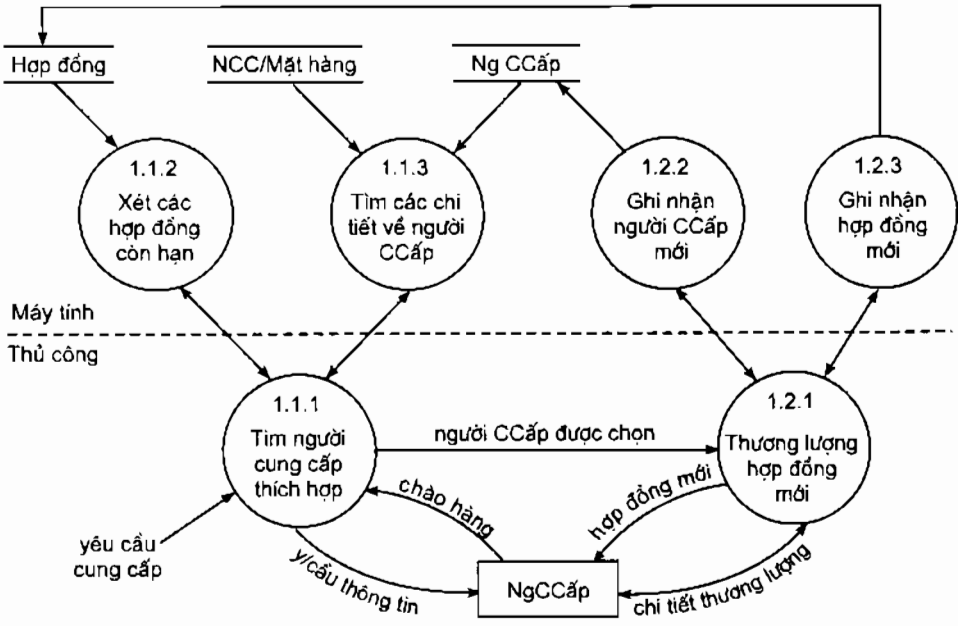
- một số chức năng có thể được quyết định chuyển trọn vẹn sang phần xử lý bằng máy tính hay phần xử lý thủ công. Tên của chúng sẽ vẫn được giữ nguyên.

- một số chức năng xét thấy cần tách một phần xử lý bằng máy tính, một phần xử lý thủ công. Vậy phải phân rã thêm một mức để tách phần xử lý bằng máy tính với phần xử lý thủ công. Chọn tên thích hợp cho các chức năng con đó.

Một thí dụ được cho trong hai hình VII.3 và VII.4 (trước và sau khi tách).



Hình VII.3 Một BLD chưa tách phần MT với phần thủ công



Hình VII.4 BLD sau khi tách phần MT với phần thủ công

(ii) Đối với các kho dữ liệu: Cũng lần lượt xem xét từng kho dữ liệu có mặt trên BLD.

- Kho dữ liệu chuyển sang phần máy tính sẽ là các kiểu thực thể tiếp tục có mặt trong mô hình dữ liệu, để sau này trở thành tệp hay cơ sở dữ liệu.
- Kho dữ liệu chuyển sang phần thủ công sẽ là:

+ các tệp thủ công (sổ sách, bảng biểu...)

+ các hồ sơ từ văn phòng.

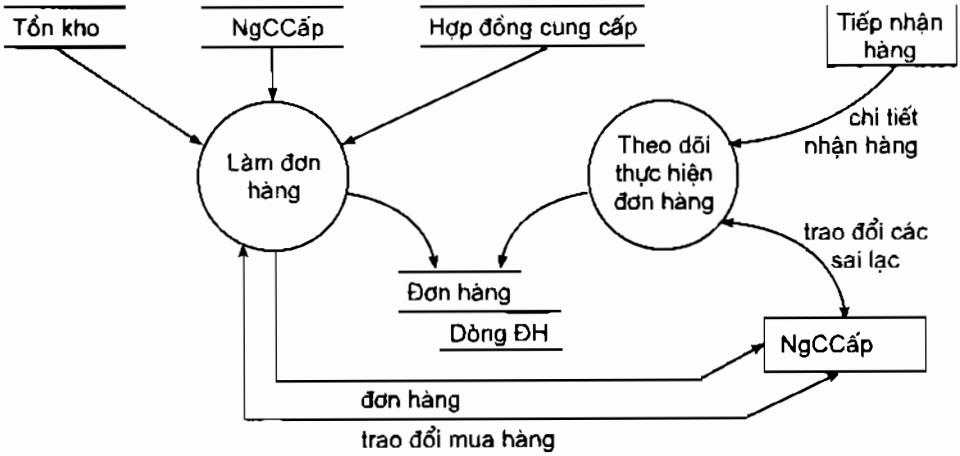
Các kiểu thực thể tương ứng với các kho dữ liệu xử lý thủ công phải loại ra khỏi mô hình dữ liệu (như vậy trong mô hình dữ liệu mới, một số kiểu thực thể biến mất, một số kiểu thực thể giảm thuộc tính). Ngược lại, do yêu cầu xử lý máy tính, một số kiểu thực thể mới có thể được thêm vào mô hình dữ liệu, chẳng hạn catalô cung cấp (hay kiểu thực thể NCC/Mặt hàng) cần phải thêm vào mô hình dữ liệu, thay cho các chào hàng trực tiếp của NgCCấp.

(iii) Chọn các phương án thể hiện khác nhau:

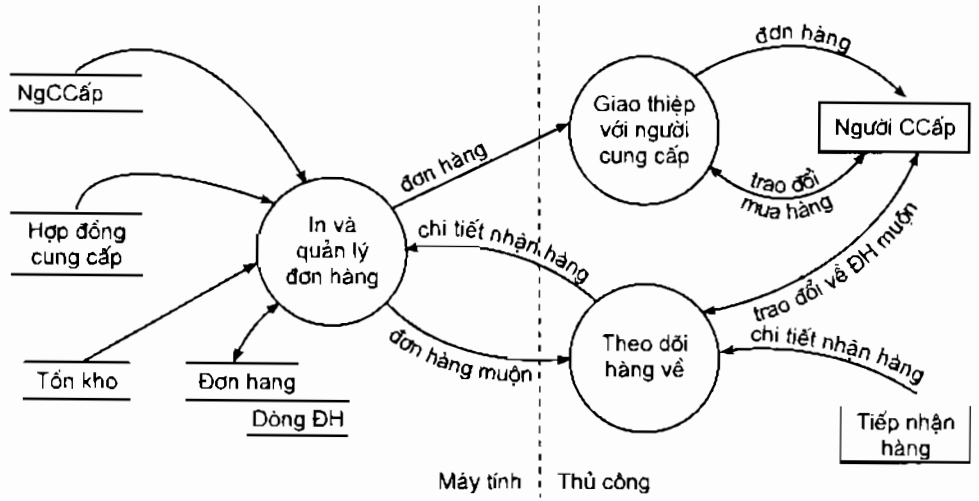
Thường thì có nhiều cách sử dụng máy tính để thực hiện các chức năng. Đặc biệt có hai phương thức xử lý thông dụng là: Xử lý theo mẻ và xử lý trực tuyến. Vậy khi thành lập BLD đã chia tách phần MT và phần thủ công, ta phải thể hiện rõ xu hướng sử dụng máy tính đã được chọn lựa.

Thí dụ: Giải quyết việc đặt hàng bổ sung cho kho hàng (Hình VII.5).

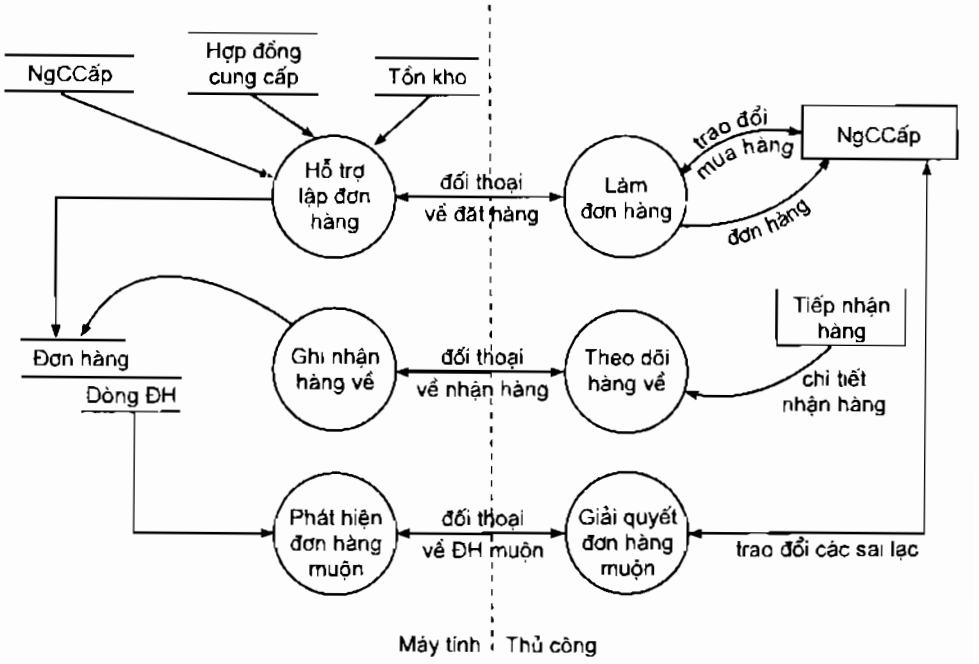
- Phương án 1 (xử lý theo mẻ): Mỗi ngày một lần kiểm tra tệp hàng lưu kho để rút ra các mặt hàng cần đặt hàng bổ sung. Mỗi tuần một lần cập nhật các hàng về và tìm các đơn hàng bị kéo dài để thúc giục người cung cấp (Hình VII.6).
- Phương án 2 (xử lý trực tuyến): Thực hiện đối thoại với máy tính bất cứ lúc nào cần. Ở đây vai trò của người là chủ yếu trong việc dẫn dắt công việc, máy tính chỉ đóng vai trò trợ giúp (Hình VII.7).



Hình VII.5 BLD chưa chia tách



Hình VII.6 BLD đã chia tách theo phương án 1



Hình VII.7 BLD đã chia tách theo phương án 2

§ 2. THIẾT KẾ CÁC NHIỆM VỤ THỦ CÔNG VÀ CÁC GIAO DIỆN NGƯỜI/MÁY

Sau khi tách phần máy tính với phần thủ công, thì hai vấn đề thiết kế đặt ra ngay:

- Thiết kế các nhiệm vụ thủ công, tức là các quy trình do con người đảm trách;
- Thiết kế các giao lưu trên biên, nghĩa là các luồng dữ liệu xuyên qua đường ranh giới giữa hai phần máy tính và thủ công. Các giao lưu này phải được thể hiện thành các loại giao diện, như là các mẫu thông tin thu thập, các tài liệu in ra từ máy tính, các màn hình.

1. Thiết kế các nhiệm vụ thủ công

a) Gom các chức năng thủ công thành các công việc và nhiệm vụ

Các chức năng nằm trong phần thủ công của BLD được đem ra nghiên cứu, mô tả lại chúng thành các quy trình, công việc. Thường thì các quy trình hay công việc đó là rất đơn giản, ta lại gom chúng vào các nhiệm vụ lớn hơn để giao cho một người hay một nhóm người thực hiện. Một lần nữa sự gom nhóm lại có thể được quyết định theo nhiều tiêu chí khác nhau:

- theo giao dịch;
- theo kho dữ liệu (và như thế người phụ trách công tác cũng phụ trách cả dữ liệu);
- theo địa điểm;
- theo thời gian xử lý;
- theo sự phân công chức trách v.v...

Nội dung và hình thức thực hiện các nhiệm vụ lại tùy thuộc vào phương thức làm việc giữa con người với máy tính, là làm việc theo mẻ hay làm việc trực tuyến. Ta đề cập các phương thức làm việc này ở các mục nhỏ tiếp sau.

b) Xử lý theo mẻ

Trong phương thức xử lý theo mẻ thì máy tính được giao giải quyết một vấn đề trọn gói, ít đòi hỏi sự can thiệp của con người, nhiệm vụ của người chỉ còn là:

- Ở đầu vào của máy tính: Thu gom thông tin, tiền xử lý bằng tay trước khi đưa vào máy tính và nhập dữ liệu vào máy tính;
- Ở đầu ra của máy tính: Kiểm tra (bằng mắt), phân phối và sử dụng các thông tin xuất từ máy tính.

So với công việc làm hoàn toàn bằng tay trước đây, thì sự làm việc (theo mẻ) với máy tính làm nảy sinh một số công việc mới cho con người:

- Kiểm soát thủ công các thông tin thu thập;
- Mã hóa các thông tin thu thập;
- Nhập liệu qua bàn phím;
- Kiểm tra các giấy tờ in ra từ máy tính;
- Phân phối các giấy tờ xuất đến các nơi sử dụng.

Có hai cách tổ chức các quy trình thủ công:

- Cách tập trung: Trung tâm máy tính phụ trách các công việc mã hóa, nhập liệu, xử lý và phân phối kết quả. Các phòng nghiệp vụ chỉ thu gom các thông tin đầu nguồn và sử dụng các kết quả ra từ máy tính.
- Cách phân tán: Đây là xu hướng hiện đại, khi mà các phòng nghiệp vụ đã được trang bị máy tính và nhất là lại cùng làm việc trên mạng. Các công việc mã hóa, kiểm tra bằng tay, sửa chữa tài liệu nhập tiến hành ở các phòng ban người dùng. Đôi khi cả việc nhập liệu trên bàn phím, kiểm tra các tài liệu xuất từ máy tính cũng giao cho các phòng ban người dùng.

Việc sử dụng máy tính đôi khi kéo theo sự cần thiết phải tổ chức lại cơ cấu của cơ quan so với khi làm việc không có máy tính trợ giúp.

c) Xử lý trực tuyến

Trong phương thức xử lý trực tuyến, thì con người đóng vai trò chủ đạo trong công việc, máy tính chỉ có nhiệm vụ trợ giúp cho con người trong việc chế biến một cách nhanh chóng một số thông tin hoặc giúp cho con người dễ dàng hơn trong việc chọn lựa các phương án hành động.

Kịch bản của một phiên làm việc trực tuyến là một sự thay đổi liên tiếp giữa sự làm việc của người và của máy tính. Để diễn tả kịch bản đó, người ta thường dùng một loại biểu đồ (một đồ thị có hướng) gọi là biểu đồ đối thoại, trong đó:

• Các nút bao gồm các loại:

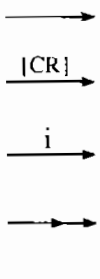
Nơi bắt đầu

Nơi kết thúc

Máy tính làm việc và dừng để chờ sự can thiệp của người

Một phần kịch bản (chương trình con) được mô tả ở một biểu đồ khác

• Các cung bao gồm các loại:



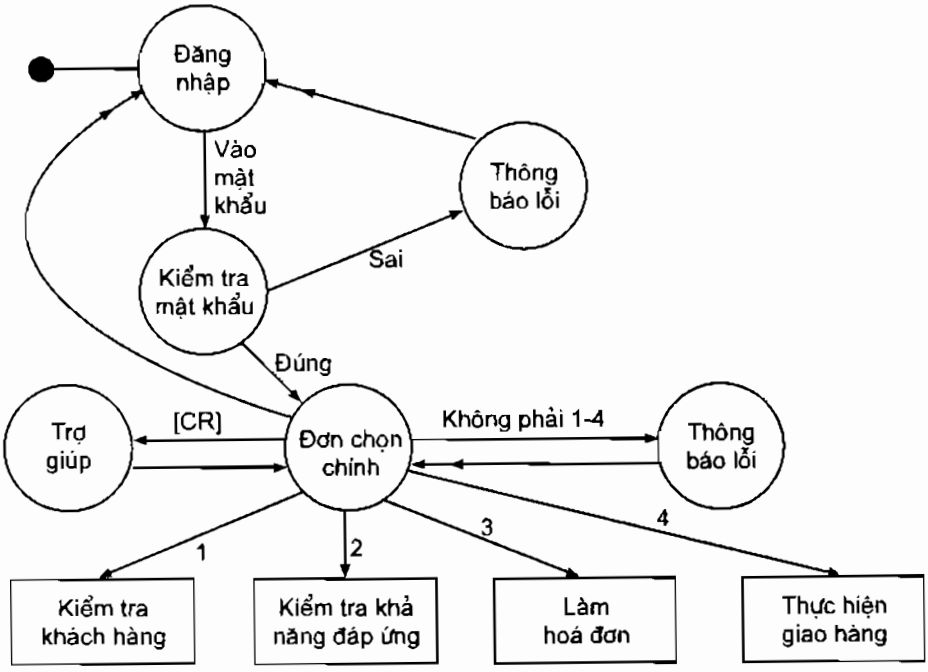
Phản ứng của người

[CR] Sự đáp lại của người bằng cách ấn phím Carriage Return (ngầm định)

i Sự chọn lựa phương án i theo đơn chọn (menu)

Sự chuyển tiếp tự động của máy tính do quá thời hạn trả lời

Thí dụ: Hình VII.8 cho một thí dụ về một biểu đồ đối thoại.



Hình VII.8 Một biểu đồ đối thoại

d) Các yêu cầu đối với việc thiết kế các thủ tục thủ công

Việc thiết kế các thủ tục thủ công phải đạt các yêu cầu sau:

- Miêu tả nội dung công việc rõ ràng: mục đích cần đạt, các bước thực hiện, yêu cầu của mỗi bước.
- Ấn định độ chính xác phải đạt.
- Ấn định mức năng suất cần thiết, mức độ khéo léo cần có.
- Hướng dẫn rõ cách xử lý khi có sai sót.

2. Thiết kế các biểu mẫu và tài liệu in

a) Các loại biểu mẫu và tài liệu in

Đó là các hình thức trình bày các thông tin để nhập vào máy tính hay xuất từ máy tính, bao gồm:

- Các biểu mẫu thu thập thông tin, như là:

  + các tờ khai,

  + các phiếu điều tra.

- Các tài liệu in ra từ máy tính, như là:

  + các bảng biểu thống kê, tổng hợp,

  + các chứng từ giao dịch (đơn hàng, hóa đơn v.v...)

b) Yêu cầu về thiết kế các biểu mẫu và tài liệu in

- Phải bao gồm đầy đủ các thông tin cần thiết.
- Các thông tin phải chính xác, và do đó phải qua kiểm tra (thông tin thu thập phải được kiểm tra trước khi đưa vào máy, thông tin xuất phải được kiểm tra trước khi chuyển giao cho nơi sử dụng).
- Phải dễ đọc, dễ hiểu, dễ sử dụng.

c) Cách trình bày các biểu mẫu và tài liệu in

Các Hình VII.9 và VII.10 cho các thí dụ trình bày một tờ khai và một chứng từ giao dịch.

WELCOME TO VIET NAM **TỜ KHAI NHẬP - XUẤT CẢNH VIỆT NAM** CHV 2000

VIET NAM ARRIVAL - DEPARTURECARD AAA 2466225

1 Họ ... tên đệm ... tên (viết chữ in hoa) ...
Family name middle name given name (in block letters)

2. ☐ Nam/ *Male* ☐ Nữ/*Female*

3. Sinh ngày ... tháng ... năm ...
Date of birth: day month year

4. Quốc tịch: ...
Nationality

5. Hộ chiếu số: ... ngày cấp: ...
Passport N° Date of issue

6. Nghề nghiệp: ...
Occupation

7. Từ /From Tới / To Số hiệu hoặc tên phương tiện vận tải / Registration N° of identification of means of transport

8. Ở Việt Nam đến ngày / Duration of stay in Vietnam:
Nơi ở hoặc cơ quan đón tiếp / Address of stay in Vietnam/ Sponsoring office or guarantor in Vietnam:

9. Mục đích nhập - xuất cảnh / *Purpose of Entry-Exit*

☐ Báo chí / *Journalism* ☐ Hội nghị / *Conference* ☐ Học tập / *Study*
☐ Đầu tư / *Investment* ☐ Du lịch / *Tourism* ☐ Thăm thân / *Family visit*
☐ Thương mại / *Business* ☐ Lao động / *Employment* ☐ Định cư / *Resettlement*
☐ Mục đích khác / *Others*

10. Họ tên, năm sinh trẻ em đi cùng hộ chiếu /*Children accompanying passport bearer (full name, date of birth)*

11. Có dấu hiệu sốt, xuất huyết, tiêu chảy, vàng da, thần kinh cấp hay không?/ *Any of the following symptoms/ syndromes: fever, haemorrhagic, diarrhea, jaundice, acute neurological syndrome?* Có / *Yes* ☐ Không / *No* ☐

12. Hành lý mang theo ... kiện, túi
Accompanied baggage ... pieces.

Hành lý gửi không cùng chuyến ... kiện
Unaccompanied baggages ... pieces

13. Ngoại hối / Foreign exchange
- Ngoại tệ trên 3.000 đô la Mỹ hoặc trên 5.000.000 đồng Việt Nam *(More than US$ 3,000 or VND 5,000,000)* Có / *Yes* ☐ Không / *No* ☐
- Vàng trên 300gr *Gold more than 300grs* Có / *Yes* ☐ Không / *No* ☐

Ghi cụ thể (nếu có) / *Declaration*

14. Hàng hóa tạm nhập-tái xuất hoặc tạm xuất-tái nhập *(Temporarily imported and re-exported goods or vice versa)* Có / *Yes* ☐ Không / *No* ☐

15. Hàng hóa phải nộp thuế. (Nếu Có thì khai báo dưới đây): Có / *Yes* ☐ Không / *No* ☐
Goods subject to duty (if yes, details in the space below):

Tên hàng hóa *Name of Goods*	Số lượng *Quantity*	Trị giá *Value*	Thuế (dành cho Hải quan) *Duties (for customs only)*

16. Tôi đã đọc phần hướng dẫn ở trang sau và cam đoan lời khai trên là đúng
I have read the instructions on the back and confirm the truth of this declaration
Ngày / *Date* _____/_____/_____
Khách ký tên / *Passenger signature*

17. Xác nhận của Hải quan *(For customs use only)*

Hình VII.9 Một tờ khai (mẫu thu thập)

BUU DIEN TP HA NOI BAN KE TONG HOP CUOC VIEN THONG
TT DICH VU KHACH HANG BĐHN THANG 11 NAM 2001
MA: ID00545A/TH29-0129

TEN KHACH HANG: ONG (/BA) NGUYEN VAN BA
DIA CHI : SO 46BIS TO 12 THANH LUONG HBT
TINH TOAN VOI BUU DIEN TP HA NOI
THEO CAC KHOAN CHI TIET SAU :

STT	CAC KHOAN CUOC PHI	SO TIEN
1	CAC KHOAN DA TINH THUE :	
2	CAC KHOAN CHUA TINH THUE :	
	CUOC THUE BAO MANG CO DINH	27 000
	CUOC DAM THOAI NOI HAT	82 720
	CUOC THONG TIN GOI DI DONG	16 528
	CUOC TRUY NHAP INTERNET VNN1269	29 960
	CONG (2)	156 203
3	THUE GTGT (2)*10%	15 620
4	TONG CONG (4) = (1)+(2)+(3)	171 823

* MOT TRAM BAY MUOI MOT NGHIN TAM TRAM HAI MUOI BA DONG

Hình VII. 10 Một chứng từ giao dịch (tài liệu xuất)

Nói chung thì mọi biểu mẫu hay tài liệu in đều gồm có ba phần chính:

- phần đầu: gồm tên tài liệu, tên cơ quan chủ quản,
- phần thân: gồm các thông tin cần thu thập hay cần xuất,
- phần cuối: gồm ngày lập tài liệu và chữ ký các người có trách nhiệm.

Ngoài ra, các tờ khai hay phiếu điều tra thường có thêm phần ghi chú hay hướng dẫn cho người khai (thường để ở mặt sau tài liệu).

Các thông tin trình bày ở phần thân thường được gom theo nhóm có liên kết chặt chẽ với nhau. Các thông tin có cấu trúc (chẳng hạn một khoản đặt hàng, gồm mã mặt hàng, mô tả mặt hàng, số lượng, đơn giá, thành tiền) thường được trình bày thành các bảng gồm nhiều cột, nhiều dòng. Tên của các cột thường phải được cân nhắc sao cho vừa rõ nghĩa vừa không quá dài để có thể viết gọn được trong ô đầu cột.

Thứ tự của các nhóm thông tin trình bày trong phần thân của tài liệu có thể được quyết định dựa trên nhiều căn cứ khác nhau, chẳng hạn:

- theo thứ tự ưu tiên,
- theo thứ tự quen dùng hay dễ điền,
- thường thì các bảng (thông tin có cấu trúc) để sau các thông tin đơn.

Về các biểu mẫu thu thập thông tin, có ba cách lấy tin:

- Khung để điền (như các mục 1, 3, 4, 5 trong Hình VII.9).
- Các trường hợp để chọn (như các mục 2, 9 trong tài liệu trên)
- Các câu hỏi để trả lời. Có hai loại câu hỏi:
  + Câu hỏi đóng, là câu hỏi mà các phương án trả lời đã được dự kiến sẵn (thực chất cũng là các trường hợp để chọn).
  + Câu hỏi mở, là câu hỏi mà sự trả lời là hoàn toàn tự do, không dự kiến trước được. Loại thông tin thu thập bằng câu hỏi mở là khó xử lý bằng máy tính, vậy nên ít dùng.

Cuối cùng về chất liệu của các biểu mẫu hay tài liệu, cũng phải được cân nhắc trên các mặt như:

- Giấy: chọn khổ giấy và loại giấy thích hợp, giấy rời hay giấy cuộn, giấy trắng hay giấy có nền in sẵn.
- Số phiên bản (một hay nhiều).
- Mầu: Mực đen trên giấy trắng thường dễ chấp nhận hơn, nhất là khi tài liệu cần được sao chụp. Tuy nhiên cũng có thể dùng mầu để:
  + phân biệt các phiên bản,
  + làm nổi bật một số thông tin trên tài liệu.

Nói chung là không nên dùng quá nhiều mầu, dùng các mầu lòe loẹt hoặc mầu đối chọi nhau vì như thế dễ gây cảm giác khó chịu cho người đọc tài liệu.

3. Thiết kế các màn hình và đơn chọn

a) Mục đích của việc sử dụng màn hình

Màn hình (cùng với bàn phím) được sử dụng để thực hiện sự đối thoại giữa người và máy.

Đặc điểm của sự tương tác kiểu đối thoại này là:

- vào/ra gần nhau (có thể nói là xen kẽ);
- thông tin cần đến là tối thiểu (có thể nói là cần đâu lấy đấy, không cần phòng dư hay tích trữ trước).

b) Yêu cầu thiết kế màn hình

- Màn hình phải sáng sủa, không lộn xộn;

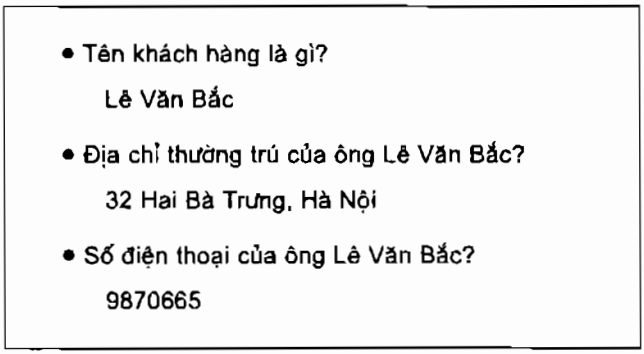
- Thể hiện rõ cái gì là được yêu cầu, cái gì là cần làm.

c) Các hình thức đối thoại

Có ba hình thức thực hiện sự đối thoại người/máy:

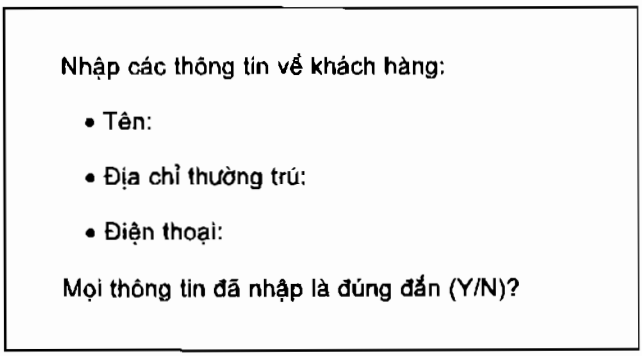
• Câu lệnh và câu nhắc: Máy hỏi hay nhắc, người đáp lại.

Chẳng hạn:



• Điền mẫu: Người dùng điền thông tin vào các chỗ trống trong một khung mẫu trên màn hình.

Chẳng hạn:



• Đơn chọn: Người dùng chọn một việc (hay một khoản) trong nhiều việc (nhiều khoản) trình bày trong một đơn chọn (menu) trên màn hình.

Chẳng hạn:

chọn một trong các việc sau

1. Vào tên
2. Vào địa chỉ thường trú
3. Vào số điện thoại:
4. Kết thúc công việc

Thay vì một danh sách các khoản trong đơn chọn, người ta có thể dùng một tập hợp các biểu tượng (icons) giàu hình ảnh và gây ấn tượng hơn.

Nếu các khoản trong đơn chọn là quá nhiều, người ta có thể dùng các đơn chọn phân cấp (chọn một khoản có thể mở ra một đơn chọn mới). Khi một đơn chọn có quá nhiều cấp để đi sâu thì việc quay lui hay ra khỏi đơn chọn cần phải được chỉ dẫn cẩn thận.

d) Các hướng dẫn cho việc thiết kế giao diện người/máy

Ngày nay các giao diện người/máy thường được thực hiện trên các màn hình dưới dạng các cửa sổ (windows). Hiện cũng đã có khá nhiều hệ thống sản sinh các giao diện, giúp ta thực hiện khá thuận lợi các giao diện đó. Tuy nhiên giao diện cần được thiết kế cẩn thận trước khi thực hiện nó.

Việc thiết kế giao diện người/máy nên thực hiện theo các bước sau đây:

- Phân loại các người dùng, theo:
  - khả năng khéo léo, tinh tế;
  - phân cấp trong tổ chức;
  - các nhóm chuyên môn mà họ tham gia.
- Mô tả nhu cầu, đặc điểm và kịch bản của mỗi loại người dùng.
- Thiết kế sự phân cấp các lệnh (command hierarchy):
  - Đưa ra danh sách các lệnh theo nhu cầu của người dùng;
  - Chỉnh đốn lại hệ thống các lệnh đó bằng cách:

+ Sắp thứ tự:

* Việc thường làm được đặt ra trước;

* Sắp theo trật tự công việc quen thuộc với người dùng.

Chẳng hạn trên một thanh đơn chọn (menu bar) ta thường gặp thứ tự như sau:

File Edit Format Calculate Monitor Window

+ Phân đoạn cho dễ nhìn:

Phân danh sách thành từng đoạn nhỏ, giúp cho người dùng có thể thâu tóm ngay sau một cái liếc mắt. Kích cỡ các đoạn được chọn dựa theo nguyên lý 7 ± 2 (Miller, 1956), hoặc 3 x 3 (Miller, 1975). Chẳng hạn ta cũng thường gặp sự phân đoạn như sau:

File
| New Drawing
| Open Drawing...
| Open Item

| Close
| Save Drawing
| Save Drawing as...
| Revert to Saved

| Page Setup
| Print...

| Quit

- Thiết kế các chi tiết tương tác:

Dựa vào các tiêu chí sau:

- Thích hợp: Chọn ngôn từ thích hợp, số bước thích hợp, các hành động thích hợp.
- Ít bước: tối tiểu hóa số bấm chuột, gõ phím, số mức các đơn chọn buông (pull down menu); Tối tiểu hóa thời gian chờ kết quả.
- Không có "thời gian chết": Chớ bỏ rơi người dùng một mình mà không biết là phải chờ cái gì và việc gì đang xảy ra.

- Lập lại: Tạo khả năng lập lại (undo) khi người dùng lầm lẫn điều gì đó.
- Người dùng không cần phải nhớ: Đừng để người dùng phải ghi nhớ điều gì (trong óc hay trên giấy) khi chuyển từ cửa sổ này sang cửa sổ khác.
- Dễ hiểu, dễ đọc: Chớ hy vọng nhiều vào các tài liệu hướng dẫn, mà nên tạo khả năng cho người dùng học ngay khi thực hành.
- Lôi cuốn và thú vị: Tạo cảm giác thoải mái cho người dùng.

§3. THIẾT KẾ CÁC KIỂM SOÁT

1. Mục đích của thiết kế kiểm soát

Mục đích của việc thiết kế các kiểm soát là đề xuất các biện pháp nhằm làm cho hệ thống đảm bảo được:

- tính chính xác,
- tính an toàn,
- tính nghiêm mật,
- tính riêng tư.

Tính chính xác của hệ thống thể hiện trước hết ở chỗ hệ thống làm việc luôn luôn đúng đắn, không đưa ra các kết quả tính toán sai lạc, không dẫn tới các quyết định kinh doanh sai lạc (chẳng hạn quyết định thực hiện giao hàng trong khi khách hàng đã có yêu cầu hủy đơn hàng, và giấy yêu cầu này lại đang tồn đọng đâu đó trong hệ thống). Các sai lạc như thế có thể bắt nguồn từ quá trình phân tích hệ thống trước đây chưa thật kín kẽ, cũng có thể do lập trình sai lỗi. Các loại kiểm định (kiểm định đơn nguyên, kiểm định tích hợp, kiểm định hệ thống) có thể giúp phát hiện các sai lỗi đó.

Tính chính xác cũng còn thể hiện ở chỗ dữ liệu dùng trong hệ thống là xác thực. Chương V đã đề cập vấn đề các ràng buộc toàn vẹn, nhằm bảo đảm tính xác thực và phi mâu thuẫn của dữ liệu. Ở đây, trong mục 2 ở dưới, ta sẽ đề cập

các biện pháp kiểm tra các thông tin thu thập và các thông tin xuất từ hệ thống, cũng nhằm bảo đảm tính xác thực của dữ liệu sử dụng.

Tính an toàn (safety) của hệ thống thể hiện ở chỗ hệ thống không bị xâm hại (hay bị xâm hại không nhiều) khi có sự cố kỹ thuật. Mục 3 dưới đây sẽ đề cập các biện pháp giảm bớt các xâm hại loại này.

Tính nghiêm mật (security) của hệ thống thể hiện ở chỗ hệ thống có khả năng ngăn ngừa các xâm phạm vô tình hay cố ý từ phía con người. Mục 4 dưới đây sẽ bàn về các loại xâm phạm đó và các biện pháp phòng ngừa.

Tính riêng tư (privacy) của hệ thống thể hiện ở chỗ hệ thống bảo đảm được các quyền truy nhập riêng tư đối với các loại người dùng khác nhau. Mục 5 ở dưới sẽ đề cập cách thực hiện tính riêng tư của hệ thống.

2. Kiểm tra các thông tin thu thập và các thông tin xuất

Để bảo đảm tính xác thực của các thông tin thu thập để đưa vào máy tính, cũng như các thông tin xuất từ máy tính, nhất thiết phải thiết lập các biện pháp kiểm tra đối với các thông tin đó.

Nơi thực hiện sự kiểm tra thường là:

- ở nơi thu thập thông tin đầu vào,
- ở trung tâm máy tính,
- ở nơi phân phối các thông tin đầu ra.

Mục đích kiểm tra là:

- Phát hiện sai lỗi,
- Chữa lỗi

Hình thức kiểm tra có thể được chọn lựa giữa nhiều phương án:

- Kiểm tra tay hay kiểm tra máy (tự động). Thường bố trí theo bảng sau:

	Nơi thu thập	TTMT	nơi phân phối
Phát hiện lỗi	tay	máy	tay
Chữa lỗi	tay	máy/tay	tay

- Kiểm tra đầy đủ hay kiểm tra không đầy đủ: Kiểm tra đầy đủ đương nhiên là tốt, song lại tốn kém, bởi vậy chỉ nên tập trung chú ý vào một số thông tin chủ yếu để kiểm tra.

- Kiểm tra trực tiếp hay kiểm tra gián tiếp;

- Kiểm tra trực tiếp là sự kiểm tra không cần dùng thông tin phụ. Chẳng hạn kiểm tra khuôn dạng của thông tin hay kiểm tra giá trị của thông tin có ở trong khoảng cho phép không đều là các kiểm tra trực tiếp.

- Kiểm tra gián tiếp là sự kiểm tra qua so đọ với các thông tin khác. Sự so đọ có thể thực hiện qua:

+ các quan hệ: = , ≠ , < , > , ≤ , ≥

+ các công thức tính toán.

Chẳng hạn: Tuổi đã khai = Năm hiện tại – Năm sinh

Thứ tự kiểm tra thường là:

- Kiểm tra trực tiếp trước,
- Kiểm tra gián tiếp sau.

Trong kiểm tra trực tiếp, ta kiểm tra lần lượt:

- Sự có mặt,
- Khuôn dạng,
- Kiểu,
- Miền giá trị.

Trong kiểm tra gián tiếp, thì cố gắng kiểm tra một thông tin khi những thông tin dùng cho việc kiểm tra đã qua kiểm tra trước đó rồi. Vì vậy ta thường kiểm tra các thông tin cần so đọ với số ít các thông tin khác trước.

Việc kiểm tra tự động (bằng chương trình máy tính) có thể cài đặt theo hai hình thức:

- Chương trình kiểm tra khi nhập thông tin theo mẻ. Cách làm này thường dùng cho các đợt thu thập thông tin với khối lượng lớn.
- Chương trình kiểm tra thông tin nhập/xuất trong làm việc đối thoại người/máy. Chương trình loại này thường được cài đặt dưới dạng các

trigger, tức là các chương trình tự kích hoạt bất cứ khi nào có sự loại bỏ, bổ sung hay cập nhật dữ liệu

3. Các sự cố làm gián đoạn chương trình và sự phục hồi

a) Nguyên nhân, tác hại và biện pháp khắc phục sự gián đoạn chương trình

Khi một chương trình bị gián đoạn vì một lí do nào đó, thì tác hại dẫn tới là:

- làm mất thì giờ, vì phải cho chạy lại chương trình từ đầu,
- làm mất hoặc sai lạc thông tin, chẳng hạn thông tin trên tệp bị sai lạc vì đang cập nhập dở dang.

Việc mất thời gian nói chung là không nghiêm trọng lắm, nhưng thông tin sai lạc là điều nguy hiểm, cần được khắc phục. Có nhiều cách để đảm bảo sự an toàn của thông tin:

- Khoá từng phần cơ sở dữ liệu: CSDL được phân hoạch thành các đơn vị để cập nhập. Các đơn vị có thể là trường, bản ghi, tệp hoặc một số phần rộng hơn của CSDL. Khi một bản sao của một đơn vị được cập nhập, thì bản gốc phải khoá lại và ngăn mọi truy nhập đến nó. Khi cập nhập kết thúc, phiên bản mới của đơn vị thay thế phiên bản cũ và sự cập nhập được hoàn thành. Nếu trong quá trình cập nhập, hệ thống bị hỏng, thì bản gốc vẫn còn nguyên vẹn.
- Các tệp sao lục: Các tệp sao lục bao gồm các tệp nhật ký và các tệp lưu. Tệp nhật ký là một tệp tuần tự chứa các bản sao (hoặc hình ảnh) của các đơn vị CSDL trước và sau khi chúng được cập nhập. Các tệp lưu gồm các bản sao toàn bộ hoặc một phần của CSDL được thực hiện theo chu kỳ. Ví dụ một bản sao một phần bảy CSDL có thể được thực hiện hằng ngày, hoặc một bản sao toàn bộ CSDL được thực hiện mỗi tuần một lần.
- Các thủ tục phục hồi: Thủ tục phục hồi là thủ tục nhằm đưa CSDL trở về trạng thái đúng đắn mà nó có ngay trước khi bị hỏng vì một sự gian đoạn chương trình. Nội dung của thủ tục phục hồi ta sẽ bàn thêm ở dưới, song cũng cần lưu ý ngay rằng sự phục hồi không phải là bao giờ cũng có ích thực sự. Điều này tuỳ thuộc vào nguyên nhân của sự gián đoạn chương trình:

- Hỏng hóc phần cứng: chẳng hạn hỏng một mạch ở trung tâm hay hỏng một thiết bị ngoại vi. Các hỏng hóc loại này nhiều khi không ngắt hẳn chương trình và để chương trình tiếp tục chạy sai lạc. Vậy phục hồi là vô ích. mà phải chữa lại máy và cho chạy lại chương trình từ đầu (có thể với một phiên bản dự trữ của tệp đang dùng).

- Giá mang của tệp có sự cố: Chẳng hạn đĩa từ bị bẩn hay bị rạch. Chương trình bị gián đoạn vì thao tác vào/ra không thực hiện được. Vậy có thể sử dụng thủ tục phục hồi.

- Hỏng hóc môi trường máy tính: Chẳng hạn mất điện, độ ẩm cao. Nếu điện ngắt hẳn, chương trình bị gián đoạn thì có thể dùng thủ tục phục phồi. Nhưng khi điện chỉ chập chờn hay máy chạy sai lạc do nóng ẩm, chương trình không bị ngắt hẳn. Phải chạy lại chương trình từ đầu, với một phiên bản dự trữ của tệp, mà không thể phục hồi.

- Hỏng hóc hệ điều hành, hiếm thấy ở các hệ điều hành một người dùng. Chương trình bị ngưng toàn bộ và định vị được, vậy có thể dùng thủ tục phục hồi.

- Thực hiện sai quy định của hệ điều hành: Chẳng hạn trong hệ điều hành nhiều người dùng, xảy ra vượt quá chỗ nhớ hay thời gian quy định, chương trình bị ngưng. Trường hợp này có thể dùng thủ tục phục hồi.

- Lỗi do lập trình sai: chẳng hạn gặp trường hợp chưa dự kiến trong chương trình. Chương trình dừng nhưng định vị được. Vậy có thể phục hồi thông tin (sau khi đã chữa lại chương trình).

- Nhầm lẫn trong thao tác: sự phục hồi là có ích.

Ngoài tác dụng của thủ tục phục hồi như đã nói ở trên, thì sự cài đặt thủ tục phục hồi còn phải được cân nhắc trên các mặt sau:

+ Thời gian bị mất do phục hồi: Phục hồi ngoài mục đích vớt vát dữ liệu, còn có mục đích tranh thủ thời gian. Nếu thời gian cho thủ tục phục hồi là lớn so với chương trình chạy bình thường, thì hà tất phải phục hồi.

+ Chương trình không bắt đầu lại được khi đã gián đoạn: Chương trình chạy theo phương thức mẻ luôn luôn có thể bắt đầu lại được. Nhưng chương trình chạy theo phương thức trực tuyến thì không thể trở lại từ đầu được. Bấy giờ thủ tục phục hồi phải được dự kiến.

+ Tính phức tạp của thủ tục phục hồi và các ràng buộc về khai thác: Nếu thủ tục phục hồi là quá phức tạp thì cần phải cân nhắc kỹ. Mặt khác cũng cần lưu ý là thủ tục phục hồi đòi hỏi phải dùng thêm nhiều tệp mới, như tệp phục hồi, tệp lưu giữ..., vậy phải đòi hỏi thêm ngoại vi.

b) Nguyên tắc hoạt động của các thủ tục phục hồi

Sao lục và phục hồi thường được các hệ thống nền (cứng và mềm) hỗ trợ. Ở đây ta đề cập chúng ở mức độ người dùng và người lập trình ứng dụng.

Chương trình có khả năng phục hồi phải thực hiện được các công việc sau:

- Khi chạy bình thường, chương trình phải lưu giữ một cách định kỳ các giá trị của một số các "biến mốc", dùng để trỏ tình trạng đang tiến triển của chương trình. Thường thì đó là chỉ số của lệnh đang được thực hiện, các giá trị trung gian nhận được... Các giá trị này đương nhiên là phải ghi lên một tệp ngoại vi mà không được để ở bộ nhớ trong để chúng còn lưu lại sau sự cố.
- Khi chạy phục hồi sau sự số gián đoạn, chương trình thực hiện các bước sau:

  + đọc các giá trị cuối cùng của các biến mốc;

  + định vị lại đầu đọc các tệp đang dùng;

  + xử lý (nếu cần) một số lô trên các tệp vận động, do có một số lô đó chưa rõ là trước khi bị gián đoạn đã được xử lý chưa;

  + khởi động lại chương trình từ chỗ bị ngắt.

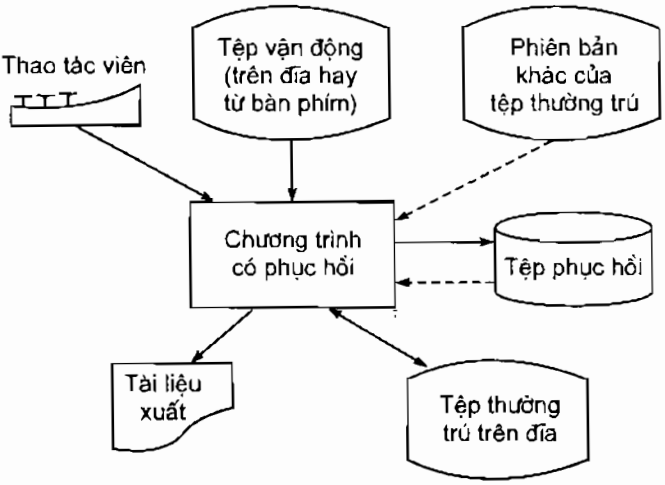
c) Cấu trúc của một chương trình có thủ tục phục hồi

Bối cảnh của chương trình có thủ tục phục hồi được mô tả trên Hình VII.11, trong đó ta hiểu:

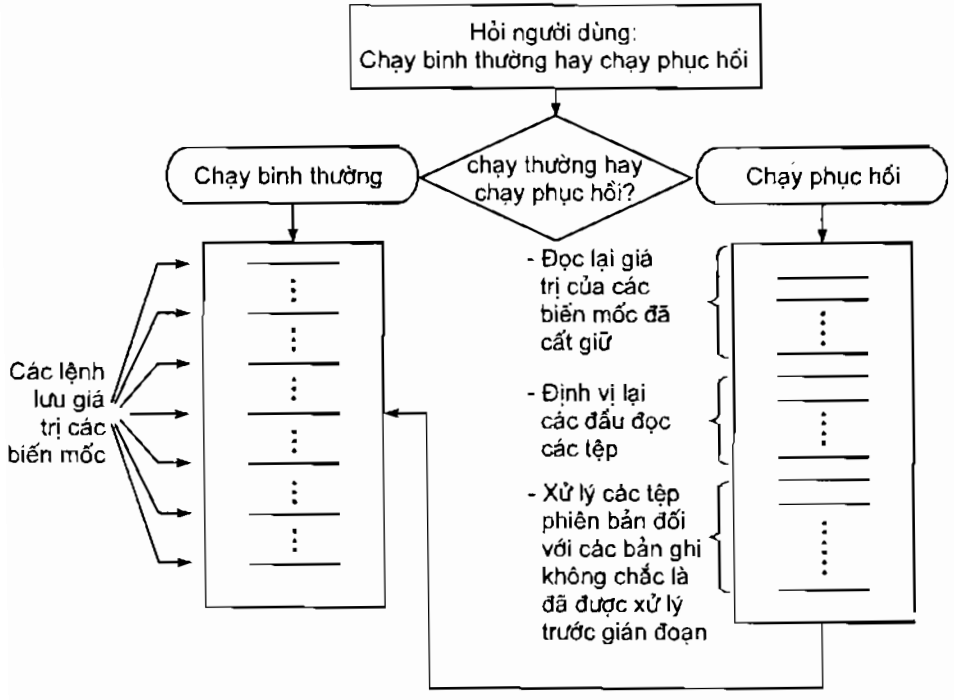
——→ là luồng thông tin trao đổi lúc chạy bình thường

----------→ là luồng thông tin trao đổi lúc chạy phục hồi.

Cấu trúc của chương trình có phục hồi cho trên Hình VII.12.



Hình VII.11 Bối cảnh làm việc của chương trình có phục hồi



Hình VII.12 Cấu trúc của chương trình có thủ tục phục hồi

Các lệnh ghi nhận các biến mốc được chèn vào chương trình ở những vị trí tùy thuộc loại chương trình:

- Loại chương trình quản lý: Thường thì mỗi chương trình quản lý được điều khiển bởi một tệp (tệp vận động, khi cập nhật, tệp thường trú khi biên tập...), gọi là *tệp dẫn*. Các biến mốc thường được ghi nhận giá trị sau n bản ghi được xử lý ($n \geq 1$). Với n =1 thì là quá đắt, thường lấy n = 20, 50, 100... Cũng có thể ghi nhận giá trị các biến mốc sau một khoảng thời gian nhất định.
- Loại chương trình tính toán khoa học kỹ thuật: Thuật toán ở đây thường khá phức tạp. Người ta ghi nhận giá trị các biến mốc sau n lần lặp, hoặc sau một khoảng thời gian nhất định.

Cách lưu giữ các biến mốc là:

- Lưu chúng lên một tệp gọi là tệp phục hồi;
- Lưu nhiều bộ nối tiếp nhau theo thời gian, thay vì một bộ, bởi vì một bộ có thể không bảo đảm (chẳng hạn có thể bị mất điện đúng lúc đang ghi các giá trị của biến mốc).

Tệp phục hồi có thể tổ chức theo hai cách:

- Tổ chức tuần tự: Lấp đầy tệp tuần tự bằng các giá trị giả, rồi ghi dần các giá trị thật. Nhờ có sự phân biệt giữa giá trị giả và giá trị thật, cho nên sau sự cố, có thể tìm được bộ giá trị được ghi cuối cùng.
- Tổ chức theo lối đu đưa: Chỉ dùng hai chỗ để ghi đủ hai bộ, luân phiên cho nhau. Đương nhiên tệp phải viết và đọc được theo lối trực tiếp (đĩa từ).

4. Các xâm phạm từ phía con người và cách phòng tránh

a) Sự xâm phạm từ con người

Con người cũng là một đầu mối gây tác hại đáng chú ý đối với hệ thống. Sự xâm phạm từ phía con người có thể là:

- Sự xâm hại vô tình, do nhầm lẫn hoặc bởi sự tò mò không có ác ý;
- Sự xâm hại cố tình, của những kẻ có mục đích rõ rệt tấn công vào hệ thống nhằm lấy cắp thông tin, làm hệ thống tê liệt, phá hoại thông tin, nhằm gây ra các quyết định sai lạc hoặc gây ra thất thoát, lãng phí tài sản.

Như vậy để bảo vệ hệ thống trước các sự xâm hại từ phía con người, ta phải nhằm vào các mục đích sau:

- Bảo vệ tính *bí mật* (confidentiality): bảo đảm thông tin không bị lộ hoặc bị khám phá bởi các cá nhân không được phép;
- Bảo vệ tính *toàn vẹn* (integrity): bảo đảm tính nhất quán của dữ liệu, đặc biệt ngăn chặn việc tạo, thay đổi bất hợp pháp hoặc phá hoại dữ liệu;
- Bảo vệ tính *khả dụng* (availability): bảo đảm sự sẵn dùng của hệ thống; người dùng hợp pháp không bị từ chối truy nhập tới thông tin và các tài nguyên một cách không đúng;
- Bảo vệ tính *riêng tư* (privacy): bảo đảm các tài nguyên không bị sử dụng bởi các cá nhân không có quyền hoặc theo các cách không hợp pháp.

b) Các điểm hở và các đe dọa từ các điểm hở

Các điểm hở là các chỗ mà một tác nhân ngoài, vô tình hay cố ý, có thể gây ra một tác động tiêu cực lên hệ thống, gây thiệt hại cho cơ quan chủ quản hệ thống. Người phân tích về kiểm soát, trước hết phải điểm diện mọi điểm hở tiềm năng đối với hệ thống mà mình phải bảo vệ. Các điểm hở có thể là:

- Mọi luồng dữ liệu đi và đến một đối tác của hệ thống;
- Mọi luồng dữ liệu cắt ngang qua ranh giới giữa phần máy tính và phần thủ công (vào/ra) trong biểu đồ luồng dữ liệu ;
- Các kho dữ liệu hoặc tệp;
- Các đường truyền trên mạng đối với các hệ phân tán.

Ở mỗi điểm hở như vậy, người phân tích kiểm soát cần phải dự đoán các mối đe dọa tiềm ẩn, thậm chí phải hình dung các kịch bản đe dọa có thể xảy ra. Từ đó có thể đánh giá mức độ đe dọa từ mỗi điểm hở, là cao, vừa hay là thấp, như thế có thể tập trung sự chú ý nhiều hơn vào những điểm nhạy cảm nhất.

c) Các biện pháp bảo mật

Có nhiều biện pháp có thể áp dụng để ngăn chặn các âm mưu xâm hại đến hệ thống. Tuy nhiên mỗi biện pháp đó đều có hạn chế. Bởi vậy ta phải phối hợp đồng thời nhiều biện pháp thì mới có thể đạt hiệu quả mong muốn. Các biện pháp có thể sắp xếp theo các mức độ, từ thô sơ đến phức tạp:

- Bảo mật vật lý: Đó là các biện pháp sử dụng các công cụ vật lý, hoặc tác động lên thiết bị, thay vì các phần mềm. Đó có thể là dùng khóa hay hệ

thống báo động để ngăn chặn và phát hiện các kẻ xâm nhập trái phép vào phòng. Cũng có thể là sự loại bỏ các ổ đĩa mềm khỏi các PC trên mạng để tránh sao chép v.v...

- Nhận dạng nhân sự: Đó là mọi hình thức (kể cả thẻ có ảnh) để nhận dạng người dùng khi vào văn phòng hay khi đăng ký nhập mạng.

- Mật khẩu: Đó là các mã xưng danh, do người dùng tự đặt và đăng ký với hệ thống, nhưng giữ kín đối với người khác, để có thể xuất trình và chứng minh sự đích thực khi hệ thống kiểm tra. Mật khẩu là biện pháp phổ biến và đắc dụng. Song sự giấu kín mật khẩu, vì lý do này hay lý do khác, thường khó kéo dài, làm cho sự kiểm soát chóng mất tác dụng.

- Mật mã: Tạo mật mã là sự biến đổi dữ liệu từ dạng nhận thức được thành dạng không nhận thức được (gọi là mật mã). Quá trình ngược lại là sự giải mã. Có nhiều phương pháp tạo mật mã (với khóa bí mật, khóa công khai...) đem lại kết quả tốt. Tuy nhiên cách làm này thường khó bảo trì và tốn kém.

- Bảo mật bằng gọi lại: Sự truy nhập thực hiện một cách gián tiếp, qua một trạm kiểm soát, tương tự cách gọi điện thoại qua tổng đài, thường được áp dụng ở các cơ quan có yêu cầu bảo mật cao. Kiểm soát được thắt chặt, song người dùng phải chịu thêm phiền phức.

- Tường lửa: Đó là một hệ thống (phần cứng + phần mềm) được đặt giữa mạng của một tổ chức, một công ty, một quốc gia (Intranet) và Internet, hay giữa các mạng có độ bảo mật khác nhau. Chức năng của tường lửa là ngăn chặn các thâm nhập trái phép (theo danh sách truy nhập đã xác định trước) và thậm chí có thể lọc bỏ những gói tin mà ta không muốn gửi đi hay nhận vào vì một lý do nào đó. Tuy nhiên tường lửa không đủ thông minh như con người để có thể đọc hiểu từng loại thông tin và phân tích nội dung tốt hay xấu của nó. Tường lửa chỉ có thể ngăn chặn sự xâm nhập của những nguồn thông tin không mong muốn, nhưng phải xác định rõ các thông số địa chỉ. Tường lửa không thể chống lại các cuộc tấn công không đi qua nó (ví dụ qua đĩa mềm). Tường lửa cũng không thể chống lại các cuộc tấn công bằng dữ liệu (data–driven attack), tức là khi một chương trình nấp dưới dạng dữ liệu (thư điện tử) vượt qua tường lửa vào mạng rồi bắt đầu hoạt động ở đây. Virus máy tính là một tní dụ như thế.

d) Phân biệt riêng tư

Phân biệt riêng tư là việc phân loại các người dùng để:

- Gán cho mỗi loại người dùng một số quyền truy nhập nhất định;
- Cho phép một số người dùng được phép ủy quyền, tức là giao quyền truy nhập cho người khác.

Không có một chuẩn thống nhất cho sự phân biệt riêng tư, song ta có thể thực hiện theo cách của SEQUEL.

Người ta dùng định danh cho mỗi người dùng làm tiền tố cho mọi tên gọi các đối tượng (dữ liệu hay chương trình), qua đó mà kiểm soát quyền truy nhập. Để thực hiện việc giao quyền ta lập hai thủ tục "ủy quyền" (GRANT) và "rút quyền" (REVOKE). Sau đây ta lần lượt xét nguyên tắc cài đặt cho hai thủ tục này:

Ủy quyền: Một người dùng có phép ủy quyền đối với một đối tượng truy nhập nào đó có thể dùng lệnh GRANT để trao quyền truy nhập đối với đối tượng đó cho người khác. Vậy trong lệnh này ta phân biệt người cho với người nhận. Người cho không nhất thiết là người chủ (đứng tên) của đối tượng truy nhập, mà chỉ là người được phép ủy quyền.

Đối với một đối tượng truy nhập là dữ liệu, thì quyền truy nhập có thể là:

- đọc (READ),
- xen một bộ (INSERT),
- loại bỏ một bộ (DELETE),
- điều chỉnh giá trị thuộc tính (UPDATE),
- đưa thêm thuộc tính (EXPAND),
- loại bỏ cả tệp (DROP),
- tạo chỉ mục (INDEX).

Đối với một đối tượng truy nhập là chương trình thì quyền truy nhập là:

- thi hành (RUN)

Dạng chung của lệnh GRANT là:

GRANT <các quyền> ON <đối tượng> TO <danh sách người dùng>
[WITH GRANT OPTION]

Nếu phần tùy chọn (trong móc vuông) được dùng, thì có nghĩa là cho phép người nhận được phép ủy quyền tiếp cho người khác.

Để cài đặt lệnh GRANT, thường phải đưa thêm vào CSDL ba quan hệ:

- QUYỀN (NgCho, NgNhận, TênQH, READ, INSERT, DELETE, EXPAND, DROP, INDEX, OPTION, UPDATE)

  trong đó READ, INSERT, DELETE, EXPAND, DROP, INDEX, OPTION là các trường Bun, còn UPDATE thì có thể nhận một trong ba giá trị: (không có gì, tất cả, một vài). Trường hợp UPDATE = một vài, thì các thuộc tính được truy nhập được ghi nhận trong quan hệ sau:

- CẬP NHẬT (NgCho, NgNhận, TênQH, Thuộctính)

  Quyền thi hành chương trình được ghi nhận trong quan hệ:

- CHẠYCT (NgCho, NGNhận, TênCT, RUN)

  trong đó RUN cũng là một trường Bun.

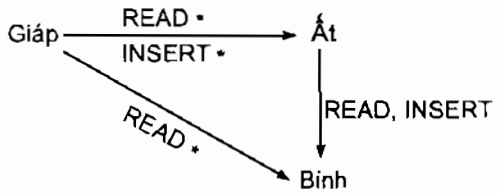
Thí dụ: Hai người dùng là Giáp, Ất lần lượt phát ra ba lệnh sau:

Giáp: GRANT READ, INSERT ON HÓAĐƠN TO ẤT WITH GRANT OPTION

Ất: GRANT READ, INSERT ON HÓAĐƠN TO BÍNH

Giáp: GRANT READ ON HÓAĐƠN TO BÍNH WITH GRANT OPTION

Tình huống sau đó có thể diễn tả trong sơ đồ sau (dấu * có nghĩa là được ủy quyền). Bính được đọc và xen vào HÓAĐƠN, song chỉ được ủy quyền tiếp với READ.



Rút quyền: Quyền truy nhập đã được giao bởi lệnh GRANT lại có thể rút lại bởi lệnh REVOKE, mà dạng chung là như sau:

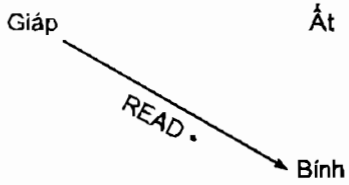
REVOKE <các quyền> ON <đối tượng> FROM <danh sách người dùng>

Một cách tự nhiên, thì nếu quyền sau khi giao lại đã được giao tiếp cho một người nhận thứ hai thì lệnh REVOKE đối với quyền đó không những phải có tác dụng dỡ bỏ đối với người nhận đầu, mà phải có tác dụng dỡ bỏ cả đối với người nhận thứ hai.

Thí dụ: Tiếp tục tình huống ở thí dụ trên kia, bây giờ Giáp phát lệnh:

Giáp: GRANT READ, INSERT ON HÓAĐƠN FROM ẤT

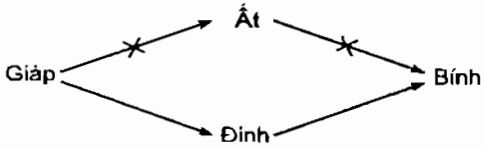
Tình huống sau đó được diễn tả trong sơ đồ sau:



Lệnh rút quyền với tác dụng lan truyền như trên rõ ràng là phải được thực hiện một cách đệ quy. Tuy nhiên việc xác định quyền nào sẽ bị rút ở một người là không đơn giản. Cần phân biệt hai trường hợp:

- Nhiều nguồn cho một quyền: Quyền đó vẫn giữ, nếu chỉ một nguồn bị rút bỏ.

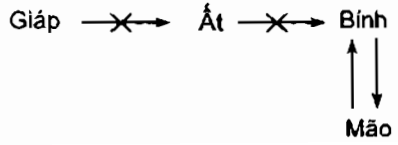
Thí dụ:



Giáp rút quyền Ất (với tác dụng lan sang Bính), song Bính còn quyền, từ nguồn của Đinh.

- Ủy quyền vòng quanh: Cần phải dỡ bỏ hết. Tuy nhiên có khó khăn, vì dường như ở đây quyền (chẳng hạn của Bính trong thí dụ sau) vẫn được thiết lập từ hai nguồn.

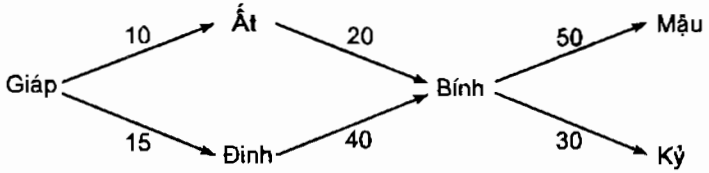
Thí dụ:



Để phân biệt quyền nào phải dỡ bỏ, và quyền nào chưa được dỡ bỏ trong quá trình rút quyền liên tiếp, người ta gán cho mỗi quyền một "con dấu" (stamp), thực chất là một giá trị số thay cho giá trị Bun nói ở trên đây cho mỗi quyền READ, INSERT, v.v... trong quan hệ QUYỀN. Con dấu này dùng để sắp thứ tự các quyền trong thời gian, đánh dấu thời điểm mà nó được thiết lập.

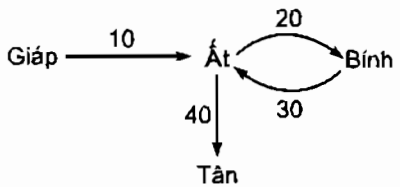
Để rút quyền, ta áp dụng quy tắc sau: Nếu A bị rút quyền, mà trước đó A đã ủy tiếp cho B trước khi A cũng nhận được quyền đó từ một nguồn khác, thì quyền của B cũng bị rút.

Thí dụ: Xét sơ đồ sau, trong đó các cung trỏ sự ủy quyền của cùng một quyền, trên cùng một đối tượng và đều trước phép ủy quyền tiếp:



Giả sử bây giờ Ất rút quyền của Bính (được giao ở thời điểm 20). Điều đó có tác dụng lan truyền là rút quyền của Kỷ được giao vào thời điểm 30, cho dù Bính vẫn còn quyền do Đinh giao vào thời điểm 40 (chú ý: 30 < 40). Trái lại quyền của Mậu vẫn giữ, vì quyền đó được xem là được giao ở thời điểm 50 và là bắt nguồn từ Đinh.

Xét tiếp trường hợp như sau:



Giả sử Giáp rút quyền Ất: cung (Giáp, Ất) bị loại, tiếp đó cung (Ất, Bính) bị loại, trong lúc đó thì cung (Ất, Tân) chưa bị loại, vì cung này được thiết lập sau khi Ất nhận cùng quyền đó từ Bính. Tuy nhiên tiếp đó cung (Bính, Ất) bị loại và rốt cục (Ất, Tân) cũng bị loại.

§4. THIẾT KẾ CƠ SỞ DỮ LIỆU

1. Mục đích

Cơ sở dữ liệu là nơi lưu giữ lâu dài các dữ liệu của hệ thống ở bộ nhớ ngoài. Các dữ liệu này phải được tổ chức tốt theo hai tiêu chí:

- *hợp lý*, nghĩa là phải đủ dùng và không dư thừa;
- *truy nhập thuận lợi*, nghĩa là tìm kiếm, cập nhật, bổ sung và loại bỏ các thông tin sao cho nhanh chóng và tiện dùng.

Ở giai đoạn phân tích, ta đã nghiên cứu dữ liệu theo tiêu chí hợp lý (đủ và không thừa). Kết quả là đã thành lập được một lược đồ dữ liệu theo mô hình thực thể/liên kết hay mô hình quan hệ. Thường gọi đó là *lược đồ khái niệm* về dữ liệu, vì nó chỉ dừng lại ở yêu cầu đủ và không thừa, mà bỏ qua yêu cầu nhanh và tiện.

Sang giai đoạn thiết kế, ta phải biến đổi lược đồ khái niệm nói trên thành *lược đồ vật lý*, tức là một cấu trúc lưu trữ thực sự của dữ liệu trong bộ nhớ ngoài. Cấu trúc này thường được chọn lựa trong số các dạng sau:

- Các tệp tuần tự,
- Các tệp tuần tự có chỉ dẫn,
- Các tệp trực truy,
- Các tệp đảo ngược,
- Các bảng băm,
- Các mạng sử dụng con trỏ,
- Các cây,
- Các quan hệ.

Mỗi cấu trúc trên đều có ưu điểm và nhược điểm riêng, buộc người thiết kế phải cân nhắc khi chọn lựa một cấu trúc thích hợp với hoàn cảnh của hệ thống mà mình đang xây dựng. Sự cân nhắc này dựa theo hai hướng nghiên cứu sau:

- Nghiên cứu các yêu cầu truy nhập dữ liệu của mỗi chức năng trong hệ thống, làm sao cho các truy nhập đó phải nhanh và tiện.

- Nghiên cứu các đặc điểm và ràng buộc của cấu hình vật lý của hệ thống (các phần cứng và phần mềm sử dụng) sao cho thích ứng được với cấu hình đó.

Vì có hai việc phải nghiên cứu như vậy, mà người ta thường tách việc thiết kế dữ liệu thành hai bước:

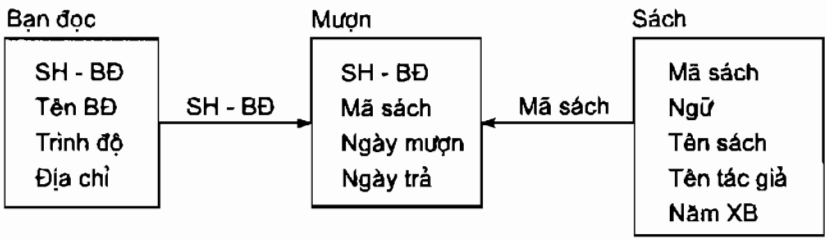
(i) Thông qua việc nghiên cứu các yêu cầu truy nhập mà biến đổi lược đồ khái niệm thành một dạng trung gian gọi là *lược đồ lôgíc* về dữ liệu.

(ii) Thông qua việc nghiên cứu cấu hình của hệ thống, đặc biệt là các ngôn ngữ lập trình, các hệ quản trị cơ sở dữ liệu đã được chọn dùng mà biến đổi lược đồ lôgíc thành lược đồ vật lý thích hợp với cấu hình đó.

2. Thành lập lược đồ lôgíc

a) Lược đồ lôgíc

Mọi cấu trúc lưu trữ vật lý, dù đó là tệp (tệp tuần tự, tệp tuần tự có chỉ dẫn, tệp trực truy...) hay là cơ sở dữ liệu (mô hình mạng, mô hình phân cấp, mô hình quan hệ) thì cũng đều tạo nên từ các đơn vị cơ sở là các bản ghi (một bản ghi cấu tạo từ nhiều trường). Vì vậy người ta chọn lược đồ lôgic, một dạng trung gian trước khi đến lược đồ vật lý, là một *cấu trúc các kiểu bản ghi*. Mỗi kiểu bản ghi là một tập hợp những bản ghi có cấu trúc trường giống nhau, thường được gọi cho gọn là một *bảng*. Bảng được biểu diễn bởi một hình chữ nhật có mang tên của bảng (viết bên ngoài) và danh sách các trường (viết bên trong). Giữa hai bảng, ta thiết lập một kết nối, diễn tả bằng một mũi tên, nếu hai bảng đó có một trường chung. Tên trường này được lấy làm nhãn cho kết nối, hơn nữa trường này phải có giá trị duy nhất cho mỗi bản ghi trong bảng ở gốc mũi tên. Một thí dụ của cấu trúc các kiểu bản ghi cho trong Hình VII.13.



Hình VII.13 Một cấu trúc các kiểu bản ghi

Bước thiết kế lôgíc xuất phát từ một lược đồ khái niệm về dữ liệu dưới dạng mô hình quan hệ đã chuẩn hóa (về 3NF). Dễ thấy rằng một mô hình thực thể/liên kết hạn chế là một cấu trúc các kiểu bản ghi, trong đó mỗi kiểu thực thể trở thành một kiểu bản ghi, còn mỗi liên kết 1-nhiều $\longrightarrow\!\!\!\!<$ trở thành một kết nối $\longrightarrow$. Cũng tương tự, một mô hình quan hệ cũng là một cấu trúc các kiểu bản ghi, trong đó mỗi quan hệ trở thành một kiểu bản ghi, còn các kết nối sẽ được thêm vào khi có mặt các khóa ngoài.

Tuy nhiên điều ngược lại thì không nhất thiết đúng: một kiểu bản ghi không nhất thiết phải hàm nghĩa một kiểu thực thể, cũng như một kiểu bản ghi không nhất thiết tương ứng với một quan hệ đã ở dạng chuẩn 3. Mục đích thiết kế lôgíc là, xuất phát từ lược đồ khái niệm ta đi tìm một cấu trúc các kiểu bản ghi phù hợp với các yêu cầu truy nhập của các chức năng xử lý trong hệ thống. Chính vì vậy, bước thiết kế lôgíc có xu hướng giật lùi so với quá trình phân tích: chấp nhận sự dư thừa, và có thể hạ chuẩn, miễn là sự lợi/hại của bước giật lùi này là đã được cân nhắc rất thận trọng.

b) Đưa thêm các thuộc tính tình thế và đánh giá các khối lượng

Trước đây, trong giai đoạn phân tích hệ thống, ta đã đưa các thuộc tính tính toán và các thuộc tính tình thế ra khỏi lược đồ dữ liệu (xem Chương IV–§2.4). Sở dĩ thế vì các thuộc tính này là dư thừa, bởi giá trị của chúng là tính toán được từ các thuộc tính khác đã có mặt trong lược đồ. Sự có mặt của chúng trong lược đồ chứa chấp một nguy cơ tiềm ẩn vi phạm tính toàn vẹn của dữ liệu.

Tuy nhiên các thuộc tính này, đặc biệt là các thuộc tính tình thế, lại là rất có ích và quen dùng trong quản lý. Chúng phản ánh trực tiếp một tình trạng nào đó của một khu vực quản lý (chẳng hạn số dư tài khoản, lượng hàng tồn kho...), thường được tra cứu luôn, mà mỗi lần dùng thì không còn phải tính toán vòng vo nữa. Chính vì vậy mà ở giai đoạn thiết kế hệ thống này, khi ta phải quan tâm đến nhu cầu truy nhập dữ liệu làm sao cho tiện và nhanh, thì ta lại phải xem lại các thuộc tính tình thế, cái nào được dùng nhiều, dùng luôn thì ta kết nạp chúng lại vào các bảng. Thậm chí, nhiều thuộc tính tình thế cùng liên quan tới một chủ đề nào đó có thể được gom lại thành một bảng mới, gọi là bảng tình thế. Cùng với việc kết nạp lại các thuộc tính tình thế này, ta phải đề xuất cac ràng buộc toàn vẹn mới (xem Ch.V–§2) để hệ thống có thể kiểm tra tính toàn vẹn của dữ liệu khi cần.

Một việc khác cần làm để chuẩn bị cho bước nghiên cứu các yêu cầu truy nhập tiếp theo là đánh giá số các bản ghi cho mỗi bảng trong lược đồ. Số này

được gọi là khối lượng của bảng và được ghi vào một ngăn bổ sung ở phía dưới bảng. Khối lượng của mỗi bảng thường là một số biến động theo thời gian. Bởi vậy ta cần tìm hiểu trong thực tế ứng dụng của hệ thống và lấy số trung bình các bản ghi của bảng trong một khoảng thời gian nào đó. Chẳng hạn trong một thư viện nhỏ, số lượng trung bình các bạn đọc là 100, số lượng trung bình các cuốn sách có trong thư viện là 2000 và số lượng trung bình các lần cho mượn sách là 5000 (trong một năm). Những khối lượng này sẽ được bổ sung vào các bảng trong Hình VII.13.

c) Nghiên cứu các yêu cầu truy nhập

Khi nói rằng cơ sở dữ liệu phải được thiết kế sao cho có thể đáp ứng các yêu cầu truy nhập một cách nhanh chóng và tiện lợi, thì sự đáp ứng này không phải hướng tới bất kỳ yêu cầu truy nhập nào, mà chủ yếu là phải hướng tới các yêu cầu truy nhập của các chức năng xử lý thông tin có mặt trong hệ thống.

Bởi vậy ta phải lần lượt xét từng chức năng trong BLD của hệ thống, để tìm trong mỗi chức năng xử lý đó có những yêu cầu truy nhập nào.

Một yêu cầu truy nhập, thể hiện một sự tìm kiếm, cập nhật, bổ sung hay loại bỏ một số thông tin trong CSDL, thường gồm có hai vế:

(i) Biết một (số) trường (tức là thuộc tính) nào đó,

(ii) Tra cứu một (số) trường khác.

Thông thường thì (các) trường biết trước (gọi là khóa tìm kiếm) và (các) trường tra cứu là không ở trên cùng một bảng, cho nên yêu cầu truy nhập được thực hiện bởi một đường truy nhập (dọc theo các kết nối liên tiếp nhau); tạo nên một dẫy các bước truy nhập.

Đối với mỗi bước truy nhập, ta cần chỉ ra bốn đặc điểm sau:

- Bảng cần được truy nhập,
- Khóa tìm kiếm,
- Trường cần tra cứu,
- Tần suất truy nhập.

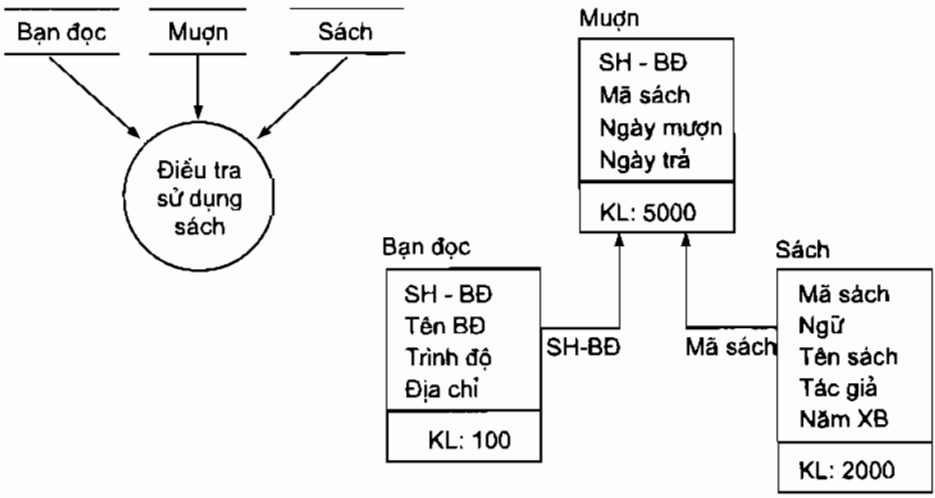
Để tính tần suất của mỗi bước truy nhập, ta áp dụng quy tắc sau:

+ Đối với bước truy nhập thứ nhất trong dẫy các bước, thì tần suất truy nhập của nó trùng với tần suất của yêu cầu truy nhập. Tần

suất này phải điều tra từ thực tế của ứng dụng, xem yêu cầu truy nhập đang xét phải thực hiện bao nhiêu lần trong một khoảng thời gian nhất định (một giờ, một ngày, một tuần, một tháng,...).

+ Đối với bước truy nhập thứ k (k > 1), thì tần suất của nó lại được tính ra từ kết quả của bước thứ k-1. Thực vậy, thông thường thì kết quả tra cứu từ bước trước lại được lấy làm khóa tìm kiếm cho bước sau. Cho nên nếu bước k-1 có tần suất là n và với một giá trị của khóa tìm kiếm, ta tìm được ở trong bước đó m bản ghi (lấy trung bình), thì bước k sẽ có 1 tần suất là n×m.

Thí dụ: Trên Hình VII.14 ta có một phần của BLD và một phần của lược đồ dữ liệu (cấu trúc các kiểu bản ghi) của một hệ quản lý thư viện.



Hình VII.14 Một hệ quản lý thư viện (trích một phần)

Giả sử khi nghiên cứu nội dung của chức năng "Điều tra sử dụng sách", ta thấy nó chứa 3 yêu cầu truy nhập dữ liệu là:

- Yêu cầu A: biết SH–BĐ, tìm Địa chỉ.
- Yêu cầu B: biết SH–BĐ, tìm NămXB các sách mà người đó đã mượn.
- Yêu cầu C: biết Mã sách, tìm Trình độ của những người đã mượn sách đó.

Yêu cầu A chỉ gồm một bước:

A/1
- bảng: Bạn đọc
- khóa tk: SH-BĐ
- tra cứu: Địa chỉ
- tần suất: 150 l/tuần

Yêu cầu B gồm hai bước:

B/1
- bảng: Mượn
- khóa tk: SH-BĐ
- tra cứu: Mã sách
- tần suất: 30 l/tuần
- mỗi lần tìm được :
- 5000/100=50 bản ghi

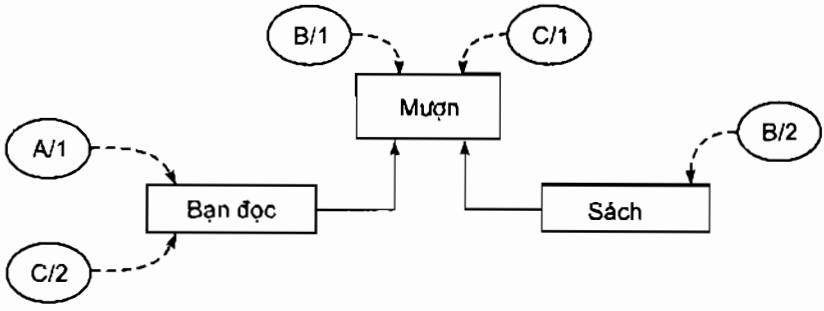
B/2
- bảng: Sách
- khóa tk: Mã sách
- tra cứu: Năm XB
- tần suất:
- 30 × 50=150 l/tuần

Yêu cầu C gồm hai bước:

C/1
- bảng: Mượn
- khóa tk: Mã sách
- tra cứu: SH-BĐ
- tần suất: 20 l/tuần
- mỗi lần tìm được :
- 5000/2000=2,5 bản ghi

C/2
- bảng: Bạn đọc
- khóa tk: SH-BĐ
- tra cứu: Trình độ
- tần suất:
- 20 × 2,5=50 l/tuần

Việc nghiên cứu các đường truy nhập như trên phải được lặp lại cho tất cả các chức năng xử lý của hệ thống. Cuối cùng có thể tổng hợp lại các kết quả vào lược đồ dữ liệu, như ở Hình VII.15.



Hình VII.15 Tổng hợp các yêu cầu truy nhập

Qua sự tổng hợp này ta có thể nhận xét nhiều điều:

- bảng nào có nhu cầu truy nhập cao;
- khóa tìm kiếm nào được sử dụng nhiều;
- các cụm trường nào thường được tra cứu cùng nhau.

Các nhận xét này cho phép ta chia cắt lại cấu trúc các kiểu bản ghi, nói ở mục tiếp sau.

d) Chia cắt lại các kiểu bản ghi

Cấu trúc các kiểu bản ghi được chỉnh sửa lại cho phù hợp với các kết quả nghiên cứu về yêu cầu truy nhập ở trên:

- Các kết nối không hề được sử dụng (tức là nhãn của kết nối không được làm khóa tìm kiếm cho một bước truy nhập nào cả) thì sẽ được dỡ bỏ (cùng với khóa ngoài tương ứng).
- Căn cứ trên sự phát hiện các cụm trường thường được tra cứu đồng thời mà thực hiện các biến đổi sau:
  + Nếu có một cụm nằm rải ra trên hai bảng, thì nên gộp hai bảng đó thành một, để bớt số bước truy nhập.
  + Nếu nhiều cụm rời nhau lại nằm trên cùng một bảng lớn, thì nên cắt bảng đó thành nhiều bảng nhỏ theo cụm, để các bảng gọn nhẹ hơn.
  + Có thể lặp lại một trường ở một bảng khác (tức là lập một bản sao của nó), nếu thấy như thế tiện tra cứu hơn.
- Nếu thấy có một bảng nào đó được truy nhập nhiều theo một khóa tìm kiếm nào đấy, thì ta nên thiết lập cho nó một đường truy nhập đặc biệt. Điều này sẽ được đề cập đối với các tệp nói ở mục dưới khi ta lập các tệp chỉ dẫn cho một tệp.

3. Thành lập lược đồ vật lý

Lược đồ vật lý là cấu trúc lưu trữ thực sự của dữ liệu trong bộ nhớ ngoài, phụ thuộc theo cấu hình của hệ thống (các ngôn ngữ lập trình, các hệ quản trị cơ sở dữ liệu...). Có hai phương án chọn lựa chính là các tệp và các cơ sở dữ liệu, theo đó mà ta phải chuyển đổi lược đồ lôgic thu được từ bước trên thành lược đồ vật lý thích hợp.

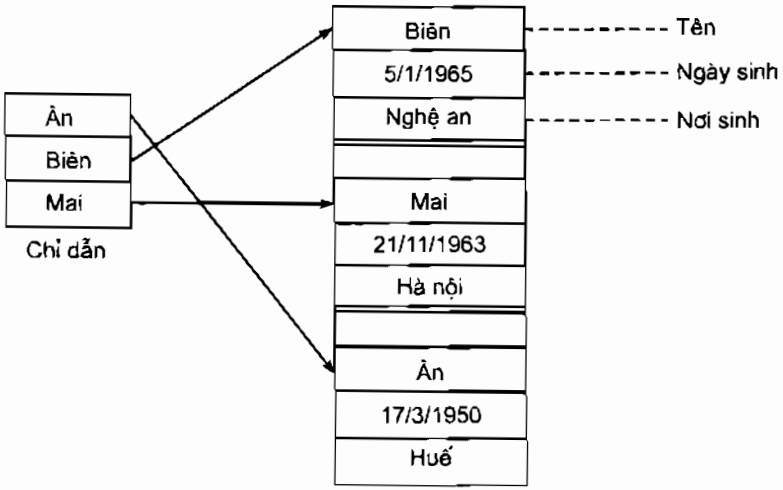
Điểm khác biệt cơ bản giữa tệp và cơ sở dữ liệu là : Đối với tệp, người dùng phải trực tiếp thành lập nó và khai thác nó, nghĩa là phải hiểu rõ và chịu trách

nhiệm trực tiếp về nó. Như thế thì chương trình và dữ liệu ràng buộc chặt chẽ với nhau. Đối với CSDL, thì sự có mặt của hệ quản trị CSDL đã cách ly chương trình của người dùng với cấu trúc lưu trữ của dữ liệu làm cho chương trình có ưu điểm là độc lập đối với dữ liệu. Đổi lại thì các ngôn ngữ định nghĩa dữ liệu, cũng như các ngôn ngữ thao tác dữ liệu cung cấp bởi các hệ QTCSDL lại phải nhúng được vào ngôn ngữ lập trình được chọn để cài đặt hệ thống, điều này không phải bao giờ cũng được thuận lợi.

a) Các tệp

Tệp là một dãy các bản ghi cùng kiểu. Mỗi bản ghi gồm nhiều trường và chiếm một số từ (hay bai) trong bộ nhớ.

Chương trình muốn tìm kiếm các bản ghi trong tệp thì phải đọc lần lượt các bản ghi từ đầu tệp cho đến khi gặp đủ các bản ghi cần tìm. Cách làm này được gọi là sự *truy nhập tuần tự*. Trường hợp cần tìm có một bản ghi thôi mà phải duyệt tuần tự như vậy thì rõ ràng là rất tốn thời gian, nhất là với các tệp lớn. Lúc đó tệp cần có khả năng *truy nhập trực tiếp*. Sự truy nhập trực tiếp được thực hiện nhờ cài thêm các *tệp chỉ dẫn*. Tệp chỉ dẫn có cùng số bản ghi như tệp chính, song đó là một tệp "nhẹ", bởi vì mỗi bản ghi của nó chỉ có hai trường, một trường là khóa chỉ dẫn, có giá trị trùng với một trường trong tệp chính được dùng làm khóa tìm kiếm cho yêu cầu truy nhập, và trường còn lại chứa con trỏ tới bản ghi tương ứng trong tệp chính. Nhờ có con trỏ lập sẵn đó mà với khóa chỉ dẫn đã biết ta có thể trực tiếp truy nhập vào bản ghi mà không cần tìm kiếm tuần tự. Hình VII.16 cho một thí dụ về tổ chức tệp chỉ dẫn đối với một tệp Nhân sự. Mỗi bản ghi của tệp Nhân sự có ba trường là Tên, Ngày sinh, Nơi sinh. Trường Tên được chọn làm khóa chỉ dẫn.



Hình VII.16 Tệp Nhân sự với chỉ dẫn

Để chuyển đổi từ lược đồ lôgíc về cấu trúc tệp, ta tiến hành như sau:

- Mỗi bảng trong lược đồ lôgic chuyển thành một tệp
- Thêm các tệp chỉ dẫn đối với các khóa tìm kiếm có tần suất sử dụng cao.

Không thể lập tệp chỉ dẫn cho mọi khóa tìm kiếm được, bởi vì có một sự hạn chế là: hệ nào cũng có một số tối đa các tệp được mở tại cùng một thời điểm. Khi một tệp chính được mở sẽ kéo theo các tệp chỉ dẫn liên quan với nó cũng được mở, làm cho số tệp mở đồng thời có thể tăng vọt, vượt ra ngoài mức hạn chế, làm cho hệ thống bị tắc nghẽn. Vậy không thể lập chỉ dẫn tràn lan được, mà chỉ hạn chế vào những nhu cầu thực sự cấp thiết mà thôi.

Hình VII.17 cho tập hợp các tệp lập từ lược đồ ở Hình VII.15

Hình VII.17 Lược đồ vật lý theo cấu trúc tệp

b) Các cơ sở dữ liệu

Cấu trúc lưu trữ vật lý phụ thuộc theo mô hình của CSDL: mô hình quan hệ, mô hình mạng hay mô hình phân cấp.

- Mô hình quan hệ : Mỗi bảng trong lược đồ lôgíc chuyển đổi trực tiếp thành một quan hệ (mỗi trường thành một thuộc tính của quan hệ). Không có cài đặt gì đặc biệt đối với các kết nối giữa các bảng, ngoại trừ sự có mặt (vốn có) của khóa ngoài. Một ngôn ngữ định nghĩa dữ liệu cho phép định nghĩa các quan hệ, và các lệnh trong một ngôn ngữ thao tác dữ liệu cho phép lưu trữ và tìm kiếm các dữ liệu dựa trên các phép toán của đại số quan hệ (thay cho sự lần dò theo các con trỏ).
- Mô hình mạng: CSDL theo mô hình mạng cũng vẫn lưu trữ dữ liệu dưới dạng các kiểu bản ghi. Ngoài ra các liên hệ cha–con giữa các kiểu bản

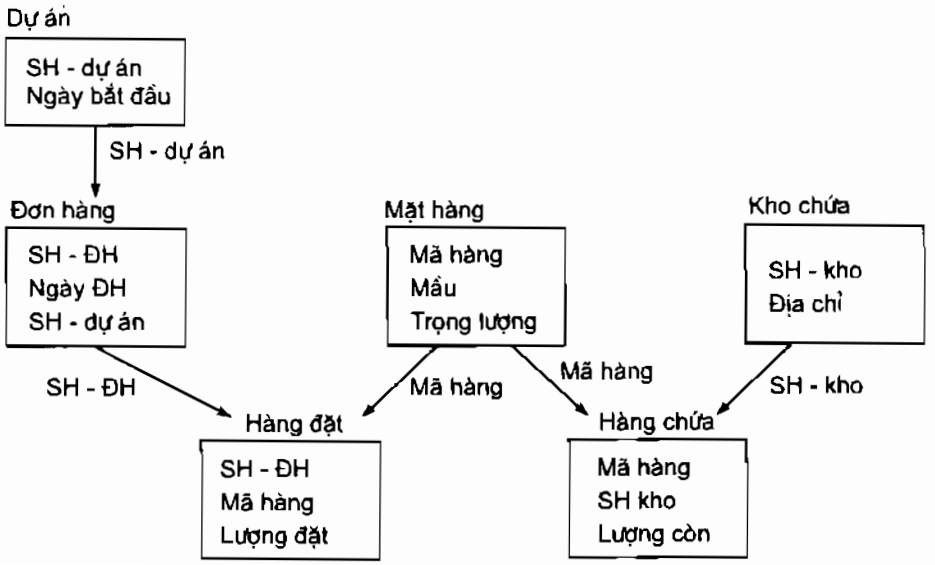
ghi (tức là các liên kết 1–nhiều) cũng được thiết lập dưới dạng các bộ con trỏ, gọi là các "set–type". Như vậy việc chuyển đổi từ lược đồ lôgic sang cấu trúc lưu giữ theo mô hình mạng cũng khá đơn giản: mỗi bảng vẫn giữ nguyên là kiểu bản ghi. Mỗi kết nối được chuyển thành một "set–type" và được gán một cái tên (chẳng hạn Set1, Set2...). Các khóa ngoài, với vai trò kết nối các bản ghi của hai bảng, khi đã có "set–type" trở nên vô dụng, cho nên được dỡ bỏ.

- Mô hình phân cấp: Cũng giống như mô hình mạng, song bắt buộc mỗi kiểu bản ghi chỉ có thể có nhiều nhất một cha. Vậy trước hết phải loại bỏ trường hợp nhiều cha trong lược đồ lôgic, bằng cách xét các liên kết 1–N từ các cha, chỉ giữ lại một cha có cơ số trung bình N nhỏ nhất, còn thì cắt đứt với các cha khác. Sau đó chuyển đổi sang cấu trúc lưu trữ (gồm các kiểu bản ghi và các set–type) như ở mô hình mạng.

Hình VII.18 cho một thí dụ về chuyển đổi một lược đồ lôgic sang cấu trúc lưu trữ theo ba mô hình khác nhau của CSDL

Hình VII.18 Chuyển đổi về cấu trúc lưu trữ

(a) Lược đồ lôgíc



(b) Mô hình quan hệ

Dự án

SH- Dự án	Ngày bắt đầu

Mặt hàng

Mã hàng	Mầu	Trọng lượng

Đơn hàng

SH - ĐH	Ngày ĐH	SH - dự án

Hàng đặt

SH - ĐH	Mã hàng	Lượng đặt

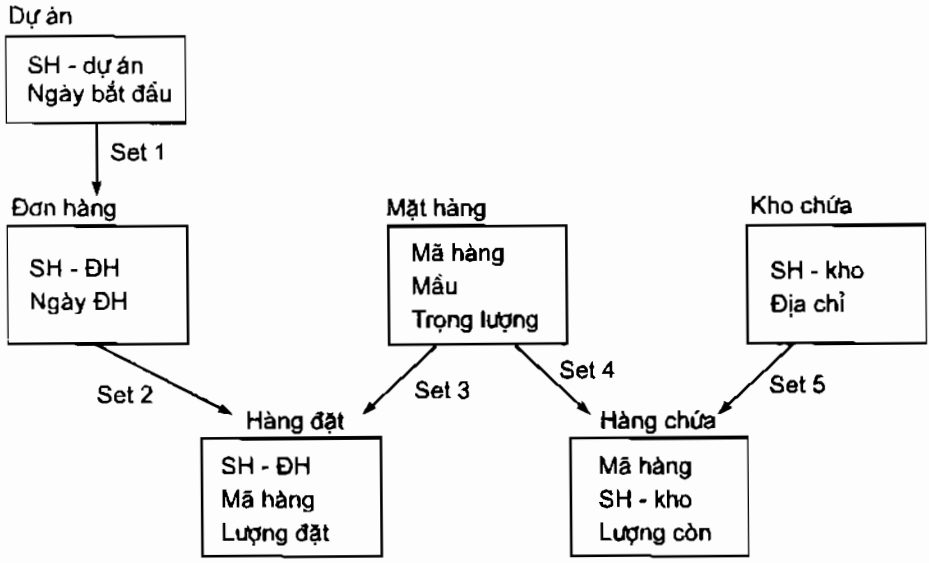
Kho chứa

SH - Kho	Địa chỉ

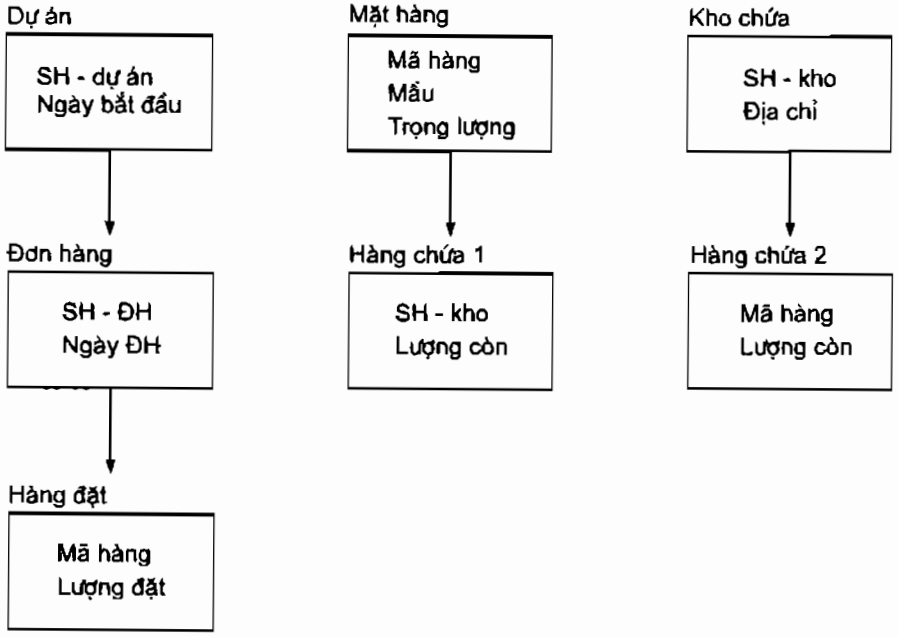
Hàng chứa

Mã hàng	SH - Kho	Lượng còn

(c) Mô hình mạng



(d) Mô hình phân cấp



§5. THIẾT KẾ CHƯƠNG TRÌNH

1. Mục đích

Tới đây thì các kết quả thu được qua các giai đoạn phân tích, thiết kế tổng thể, và thiết kế chi tiết (về các giao diện, kiểm soát và cơ sở dữ liệu) dù là khá phong phú, nhưng vẫn còn là chưa đủ để có thể chuyển sang lập trình được. Các yếu tố còn thiếu là:

- Các chức năng xuất hiện trong các BLD chỉ là các chức năng lôgíc (thuộc lĩnh vực bài toán) mà chưa có các chức năng phù trợ cần thiết, như là:

  + các chức năng đối thoại với người dùng,

  + xử lý lỗi,

  + xử lý vào/ra,

  + tra cứu CSDL,

  + các chức năng điều hành (nhằm liên kết các chức năng khác).

- Các liên quan giữa các chức năng trong BLD chỉ là các chuyển giao dữ liệu mà không phải là các chuyển giao điều khiển (tức là chuyển giao sự thực hiện, khi thi hành). Thế mà một đặc trưng không thể thiếu trong một chương trình là đặc trưng điều khiển (sự tuần tự, chọn, lặp và đặc biệt là lời gọi giữa các chương trình con). Đặc trưng này chưa hề có trong các BLD.

Vì các thiếu sót này mà các BLD thu được từ giai đoạn phân tích còn phải được biến đổi, bổ sung thêm chi tiết, thì mới trở thành đầu vào thực sự cho việc lập trình được. Vì vậy phải có thêm một giai đoạn thiết kế chi tiết, đó là thiết kế chương trình. Chú ý rằng đây cũng chỉ là một giai đoạn của thiết kế, nhằm đưa ra các quyết định về cài đặt, chứ chưa phải là cài đặt, chưa phải là lập trình thực sự.

Đầu vào cho việc thiết kế chương trình là BLD của từng hệ thống con (đã xác định trong thiết kế tổng thể) cùng với các quyết định về giao diện, kiểm soát và CSDL đã được chọn trong các bước thiết kế chi tiết trước đây.

Đầu ra của thiết kế chương trình là một miêu tả về nội dung các chương trình sẽ được cài đặt, bao gồm:

- Một lược đồ chương trình (LCT) cho mỗi hệ thống con. Lược đồ chương trình được trình bày dưới dạng một đồ thị có hướng thường gọi là lược đồ cấu trúc (structure chart), trong đó:
  + mỗi nút là một môđun chương trình,
  + mỗi cung là một lời gọi (lời gọi của môđun ở gốc đối với môđun ở ngọn của cung).
- Đặc tả nội dung của từng môđun trong LCT.
- Phân bổ các môđun trong LCT thành các chương trình (hay môđun tải).

Sau đây ta lần lượt xét các công việc để thực hiện mục đích nói trên.

2. Lược đồ cấu trúc

Lược đồ cấu trúc (gọi là *lược đồ chương trình* cũng được) là một biểu diễn dưới dạng đồ thị của một tập hợp các môđun cùng với các giao diện giữa các môđun đó (bao gồm sự chuyển giao điều khiển và chuyển giao dữ liệu).

a) Các môđun chương trình

Trong định nghĩa của lược đồ cấu trúc nói trên, thì môđun có thể được hiểu là:

- một chương trình con (PROCEDURE, FUNCTION, SUBROUTINE...)
- có khi chỉ là một cụm câu lệnh nằm trong chương trình.

Nhiều ngôn ngữ lập trình có các UNIT, CLASS, OBJECT. Đó thực chất là các nhóm môđun chương trình (thường gọi là các phương thức) tập hợp xung quanh một cấu trúc dữ liệu.

Nói chung thì môđun có bốn thuộc tính cơ bản sau:

- Vào/Ra:
  + Cái vào: nó nhận được thông tin gì từ chương trình gọi nó
  + Cái ra: nó trả lại thông tin gì cho chương trình gọi nó.
- Chức năng: Ấy là ánh xạ từ cái vào thành cái ra.
- Cơ chế: Ấy là phương thức cụ thể để biến đổi cái vào thành cái ra.
- Dữ liệu cục bộ: Ấy là các chỗ nhớ hay cấu trúc dữ liệu dùng riêng cho nó.

Vào/Ra + Chức năng tạo thành đặc trưng bề ngoài của một môđun. Còn Cơ chế + Dữ liệu cục bộ tạo thành đặc trưng bên trong của một môđun. Phương pháp thiết kế có cấu trúc lưu ý trước hết đến đặc trưng bề ngoài của môđun và lùi việc xem xét các đặc trưng bên trong của môđun lại sau.

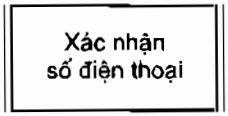
Ngoài bốn thuộc tính cơ bản trên, môđun chương trình còn có một số thuộc tính khác cũng cần lưu ý khi thiết kế :

- Tên của môđun, cho phép dùng tên để gọi môđun (tức là huy động nó).
- Chỗ chiếm của nó (ở trong văn bản chương trình, hay trong bộ nhớ).

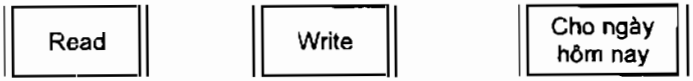
b) Các yếu tố hợp thành LCT

(i) *Các môđun:* biểu diễn bởi một hộp chữ nhật với tên môđun ở bên trong. Tên môđun phản ánh tóm tắt chức năng của môđun.

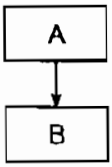
Nếu là môđun đã định nghĩa sẵn (trong hệ thống, trong thư viện chương trình), thì các cạnh bên được vẽ kép.



Ta thường gặp các môđun định nghĩa sẵn trong hệ điều hành, như là:



(ii) *Kết nối các môđun:* Các môđun có thể được kết nối với nhau bằng các lời gọi, diễn tả bởi một mũi tên (cung)



– Môđun A gọi môđun B;

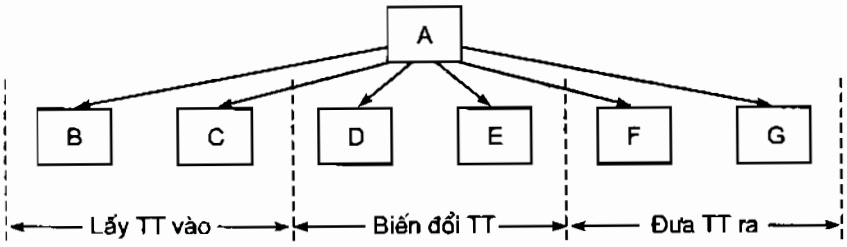
– Môđun B thực hiện chức năng của mình rồi trả điều khiển cho A ở vị trí sau lời gọi.

Trong cách biểu diễn các lời gọi bằng các cung giữa các môđun như trên thì ta có thể hiểu thêm như sau:

+ Vị trí của lời gọi (trong môđun A), cũng như số lần gọi đều không được chỉ rõ. Điều đó có nghĩa là ở giai đoạn lập LCT thì các chi tiết này có thể tạm bỏ qua, để giảm bớt sự phức tạp.

+ Trật tự trước sau của nhiều lời gọi xuất phát từ cùng một môđun được thể hiện bởi trật tự từ trái qua phải.

Như vậy trong dãy các môđun được gọi bởi cùng một môđun thường phân biệt ba khu vực: bên trái là khu vực gồm các môđun lấy các thông tin vào, ở giữa là khu vực gồm các môđun xử lý các thông tin, và bên phải là khu vực gồm các môđun đưa các thông tin ra (Hình VII.19).

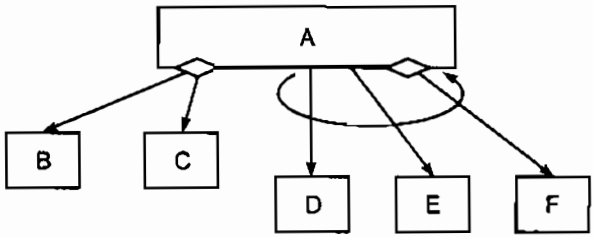


Hình VII.19 Trật tự các lời gọi

+ Đôi khi người ta thể hiện phép chọn giữa một số lời gọi bằng một hình thoi nhỏ, và thể hiện phép lặp một nhóm lời gọi bằng một mũi tên vòng. Chẳng hạn trên Hình VII.20, ta hiểu là môđun A:

– gọi môđun B hoặc môđun C (tùy thuộc một điều kiện nào đó)

– và tiếp đó thực hiện vòng lặp các lời gọi tới D, E và thỉnh thoảng F.

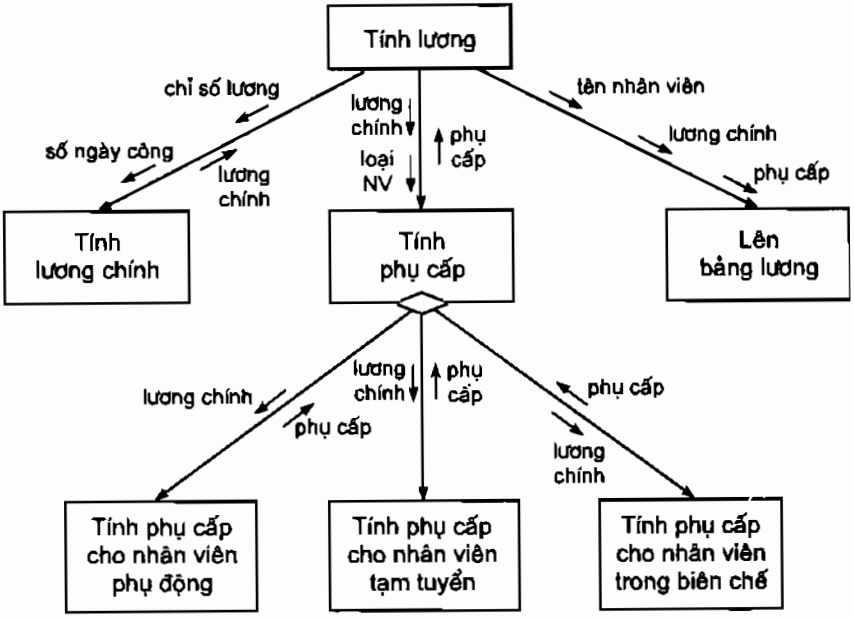


Hình VII.20 Chọn và lặp các lời gọi

(iii) *Thông tin trao đổi giữa các môđun:*

Các thông tin được gửi đi kèm với lời gọi (các tham số) và thông tin trả về sau khi thực hiện lời gọi (trả lời) được thể hiện bằng các mũi tên nhỏ vẽ dọc theo cung biểu diễn cho lời gọi, có kèm theo tên của thông tin.

Một thí dụ về LCT được cho trong Hình VII.21.



Hình VII.21 Một lược đồ cấu trúc

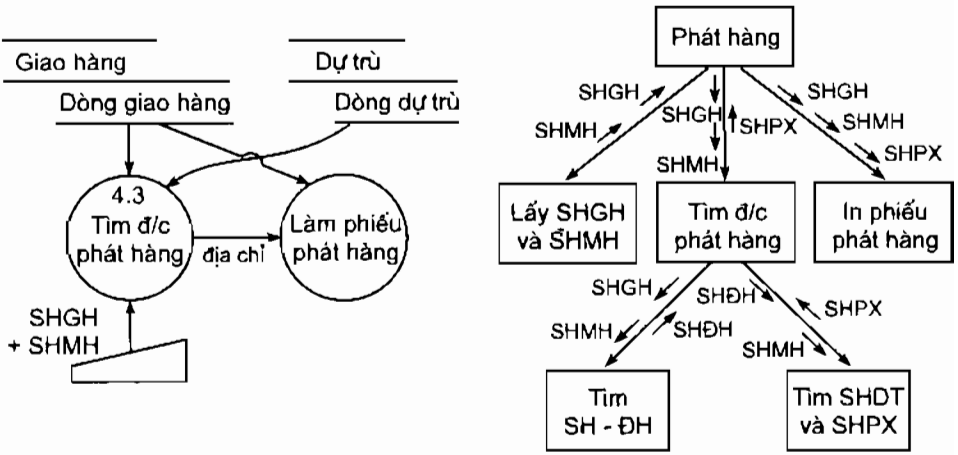
3. Cách chuyển đổi BLD thành LCT

a) Yêu cầu chung

Đối với mỗi BLD của một hệ thống con, ta phải lập một LCT tương ứng. LCT này phải đạt các yêu cầu sau:

- Nhiệm vụ của mọi chức năng xử lý trong BLD phải được chuyển hết vào các môđun chương trình của LCT. Đó có thể là sự chuyển đổi mỗi chức năng xử lý thành một môđun chương trình, mà cũng có thể là sự phân tán hay tổ hợp các chức năng xử lý vào các môđun chương trình khác nhau.
- Phải thêm các môđun vào/ra (giao diện với người dùng hay truy nhập CSDL), và đặc biệt là thêm các môđun điều khiển làm nhiệm vụ dẫn dắt quá trình xử lý. Trong số các môđun điều khiển đó phải có một môđun đặt ở gốc của LCT, thường được gọi là môđun chính (main).
- Thiết lập các lời gọi (kèm với các thông tin chuyển giao) giữa các môđun, phản ánh được quá trình thực thi của chương trình.

Trước khi xem xét các cách thức chuyển đổi BLD thành LCT, ta hãy xét một thí dụ cụ thể. Hình VII.22 cho một BLD nhỏ (đó là một phần của hệ thống con 2.2 của hệ CƯVT đã cho trên Hình VII.2) và LCT được chuyển đổi từ BLD đó. Qua đây ta thấy sự khác biệt về cấu trúc của hai loại công cụ biểu diễn này.

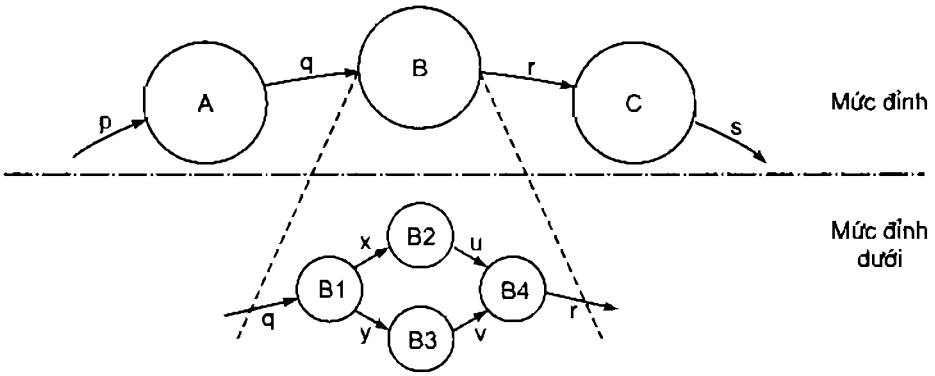


Hình VII.22 Một BLD và LCT tương ứng

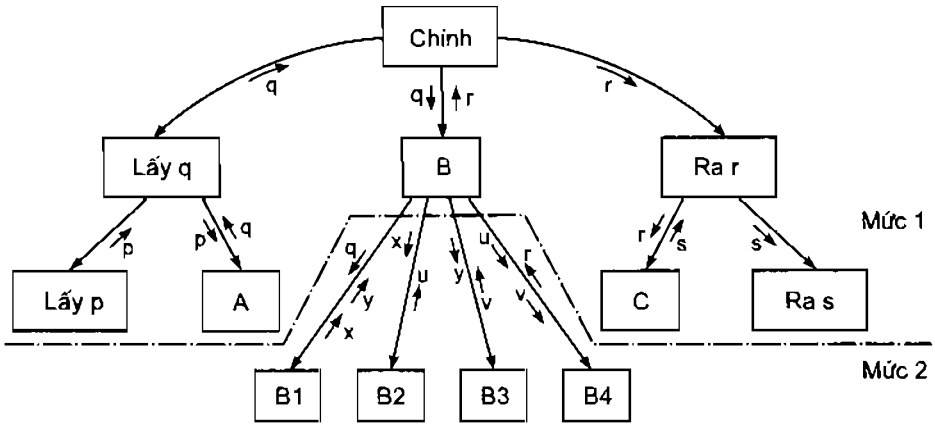
b) Triển khai trên xuống

Một LCT với các môđun và các lời gọi của nó tạo nên một sự phân cấp. Bởi vậy cách thành lập LCT tốt nhất là triển khai dần từ trên xuống. Sự triển khai trên

xuống đó lại có thể kết hợp chặt chẽ với sự phân mức các BLD mà ta thu được từ giai đoạn phân tích. Chẳng hạn BLD mức đỉnh có dạng như trong Hình VII.22 (a), trong đó chức năng B triển khai thêm một mức trở thành một BLD mức dưới đỉnh với các chức năng B1, B2, B3, B4. Căn cứ trên BLD mức đỉnh (ở đây có dạng tuyến tính), ta dễ dàng lập được phần chóp của LCT (xem Hình VII.22 (b)), rồi sau đó triển khai B với các lời gọi tới các môđun mới B1, B2, B3, B4.



Hình VII.22 (a) BLD mức đỉnh, với định nghĩa của B



Hình VII.22 (b) LCT tương ứng, triển khai theo hai mức

Cách làm như trên là khá đơn giản, song LCT thu được thường là quá rườm rà. Vả lại, khi gặp một BLD (ở một mức nào đó) có dạng phức tạp, thì việc chuyển nó thành LCT không phải là dễ như ở thí dụ trên. Các trường hợp phức tạp thường thể hiện dưới hai dạng điển hình: cấu trúc chế biến (transform

structure) và cấu trúc giao dịch (transaction structure). Do đó có hai hướng thiết kế khác nhau sẽ được trình bày ở các mục nhỏ tiếp sau.

c) Thiết kế hướng theo chế biến

Thiết kế hướng theo chế biến (transform–centered design) áp dụng cho trường hợp BLD có nhiệm vụ chế biến một số thông tin lấy từ một số nguồn phát, thành một số thông tin gửi tới nơi nhận.

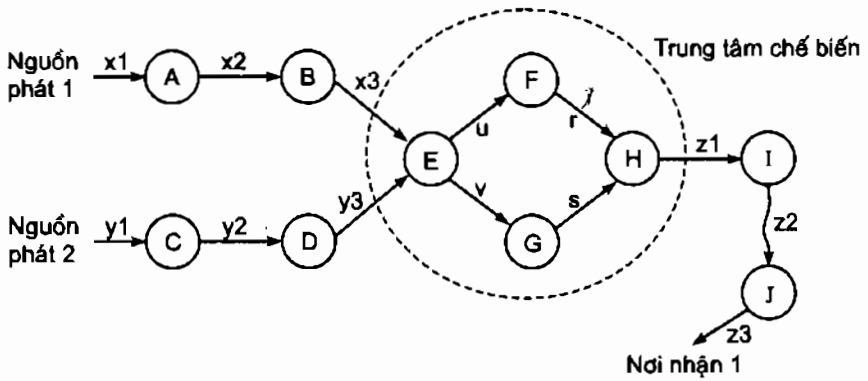
Các bước thực hiện là như sau:

(i) Trước hết xác định nhóm các chức năng góp phần thực sự và chủ yếu vào nhiệm vụ chế biến thông tin của BLD. Gọi nhóm này là *trung tâm chế biến*. Phần còn lại của biểu đồ gồm:

+ Các tuyến lấy thông tin vào, mỗi tuyến vào này xuất phát từ một nguồn phát, dẫn thông tin qua một số bước sơ chế (nghĩa là những xử lý "nhẹ" như là biên tập, sắp xếp,...) để cuối cùng trở thành một đầu vào của trung tâm chế biến.

+ Các tuyến chuyển thông tin ra, mỗi tuyến ra này xuất phát từ một đầu ra của trung tâm chế biến, đưa thông tin đi qua một số bước gia công (nghĩa là những xử lý nhằm chỉnh sửa chút ít, như là chuyển thành dạng dễ hiểu, dễ dùng hơn) để đến một nơi nhận.

Một thí dụ về sự phân biệt trung tâm chế biến với các tuyến vào/ra cho trên Hình VII.23 (a).



Hình VII.23 (a) BLD có một trung tâm chế biến

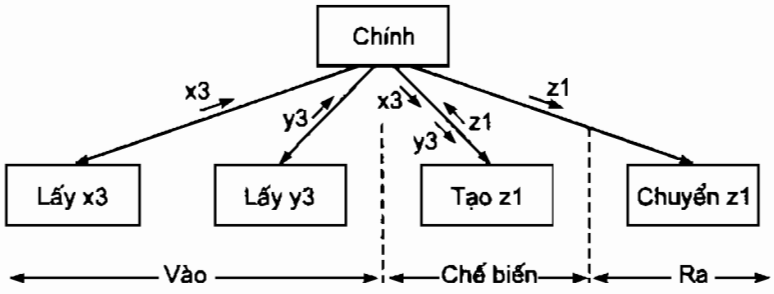
(ii) Vẽ hai mức cao nhất của LCT, trong đó môđun đỉnh gọi đến:

+ một môđun lấy thông tin cho mỗi tuyến vào,

+ một môđun tạo thông tin cho trung tâm chế biến,

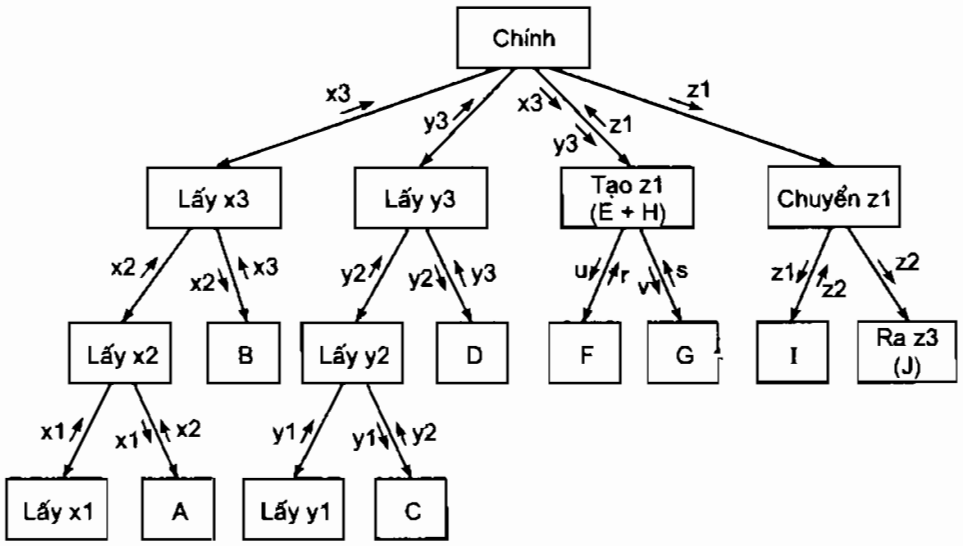
+ một môđun chuyển giao thông tin cho mỗi tuyến ra.

Kết quả thực hiện bước này đối với BLD cho trong Hình VII.23 (a) được trình bày trong Hình VII.23 (b).



Hình VII.23 (b) Hai mức đầu của LCT chế biến

(iii) Triển khai mỗi môđun ở mức 2 (vào, ra hay chế biến) thành các môđun ở các mức thấp, cho đến khi chuyển hết mọi nhiệm vụ xử lý của BLD vào LCT. Kết quả cuối cùng của thí dụ trên là LCT ở trong Hình VII.23 (c)



Hình VII.23 (c) LCT đã bao gồm hết nội dung của BLD

d) Thiết kế hướng theo giao dịch

Đặc trưng của cấu trúc giao dịch là có một chức năng phân loại cho phép xác định loại của dữ liệu vào, để rồi cứ mỗi loại sẽ cung cấp một cách xử lý (hay dịch vụ) riêng. Thiết kế hướng theo giao dịch (transition–centered design) áp dụng cho trường hợp BLD thể hiện một cấu trúc giao dịch như vậy.

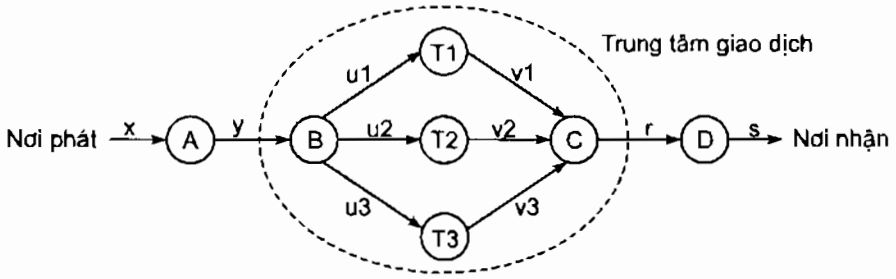
Các bước thực hiện là như sau:

(i) Trước hết xác định *trung tâm giao dịch* trong BLD, bao gồm chức năng phân loại dữ liệu vào cùng với các chức năng tham gia vào các quá trình xử lý cho từng loại dữ liệu. Phần còn lại của biểu đồ gồm:

+ Một tuyến lấy thông tin vào, dẫn thông tin từ một nguồn phát, qua một số chức năng sơ chế để đến chức năng phân loại thuộc trung tâm giao dịch.

+ Một (hay một số) tuyến đưa thông tin ra, dẫn thông tin đưa ra từ các xử lý theo trường hợp, qua một số chức năng gia công thêm để tới một (hay một số) nơi nhận.

Một thí dụ về phân biệt trung tâm giao dịch với các tuyến vào/ra cho trên Hình VII.24 (a)



Hình VII.24(a) BLD có một trung tâm giao dịch

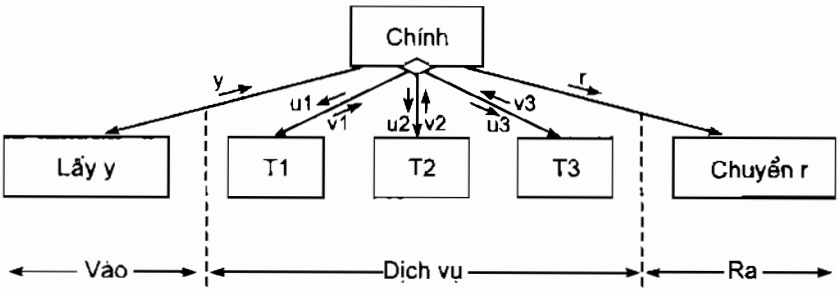
(ii) Vẽ hai mức cao nhất của LCT, trong đó môđun đỉnh gọi đến:

+ một môđun lấy thông tin vào cho tuyến vào,

+ một môđun xử lý (dịch vụ) cho mỗi trường hợp của dữ liệu vào; các lời gọi này được kết nối qua một phép chọn,

+ một môđun chuyển giao thông tin cho mỗi tuyến ra.

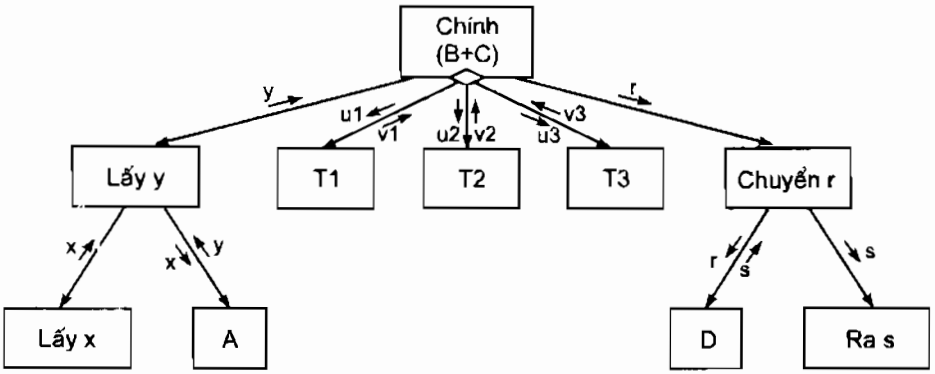
Kết quả thực hiện bước này cho BLD ở Hình VII.24 (a) được trình bày trên Hình VII.24 (b)



Hình VII.24 (b) Hai mức đầu của LCT giao dịch

(iii) Triển khai mỗi môđun ở mức 2 (vào, ra hay dịch vụ) thành các môđun ở các mức thấp, cho đến khi chuyển hết mọi nhiệm vụ xử lý của BLD vào LCT.

Kết quả cuối cùng với thí dụ trên là LCT ở trong Hình VII.24 (c)



Hình VII.24 (c) LCT đã bao gồm hết nội dung của BLD

Chú thích: Hai phương pháp lập LCT theo chế biến và theo giao dịch nói trên là hai cách làm vận dụng cho các tình huống khác biệt. Nói chung thì cấu trúc chế biến, theo các mức độ phức tạp khác nhau, là hay gặp hơn. Tuy nhiên trong một BLD lớn, thường thì cả hai cấu trúc đều có thể xuất hiện ở những vùng này vùng nọ. Lúc đó ta phải vận dụng cả hai phương pháp, chỗ thì hướng theo chế biến, chỗ thì hướng theo giao dịch.

4. Chất lượng của LCT

LCT lập được từ các cách làm trình bày ở trên chưa nên xem là dạng cuối cùng để chấp nhận. Trái lại ta chỉ có thể xem đây là phác thảo bước đầu của thiết

kế môđun. Ta còn phải tinh chỉnh nó lại, bằng cách gộp, tách hay san sẻ lại nhiệm vụ giữa các môđun để đạt được các tiêu chuẩn về chất lượng nói ở sau đây.

a) Sự tương liên

Sự tương liên (coupling) nói lên mức độ ảnh hưởng lẫn nhau giữa các môđun. Đương nhiên không thể không có tương liên giữa các môđun trong cùng một LCT. Tuy nhiên để có một LCT tốt, sự tương liên giữa các môđun phải càng lỏng lẻo, càng đơn giản càng hay. Có nhiều loại tương liên, ta hãy xét vài loại chính:

- Tương liên nội dung: Đó là trường hợp môđun này can thiệp nội dung của môđun khác. Ví dụ một môđun làm thay đổi nội dung (các lệnh) của một môđun khác, rẽ nhánh sang một môđun khác, hay sử dụng dữ liệu cục bộ (không phải dữ liệu phản hồi) của môđun được gọi. Tương liên nội dung là tương liên xấu nhất, cần phải loại bỏ.
- Tương liên điều khiển: Đó là trường hợp một môđun này chuyển một thông tin điều khiển cho một môđun khác. Thông tin điều khiển là thông tin dùng vào việc điều khiển quá trình, chứ không là chất liệu của các xử lý. Chẳng hạn các cờ, trạng thái, giá trị Bun dùng để tính điều kiện của một phép chọn v.v... Tương liên điều khiển vi phạm nguyên tắc che giấu thông tin. Bởi vì khi một môđun chuyển một thông tin điều khiển cho một môđun bị gọi, thì như vậy nó đã phải biết phương thức bên trong của môđun bị gọi. Một thay đổi đối với môđun bị gọi sẽ kéo theo sự thay đổi đối với môđun gọi. Vậy tương liên điều khiển cũng nên tránh. Tuy nhiên ta không thể tránh tương liên điều khiển một cách triệt để được, vì nhiều khi nó vẫn có ích, vẫn cần thiết.
- Tương liên dữ liệu : Đó là trường hợp hai môđun trao đổi dữ liệu cho nhau. Đó là loại tương liên cần phải được chấp nhận. Song sự trao đổi dữ liệu cũng phải càng đơn giản càng tốt, nghĩa là:
  + ít dữ liệu tốt hơn là nhiều dữ liệu trao đổi;
  + nếu có cách trao đổi chuẩn (chẳng hạn chuyển giao tham số), thì nên dùng nó;
  + nên chuyển dữ liệu thật, hơn là chuyển con trỏ đến dữ liệu.

b) Sự cố kết

Sự cố kết (cohesion) nói lên sự gắn bó giữa các phần bên trong một môđun. Về phương diện này, thì một môđun nên thực hiện một chức năng lôgíc duy

nhất hơn là nhiều chức năng khác nhau. Môđun càng cố kết, thì chức năng của nó càng dễ thấy, lôgic của nó rành mạch, do đó dễ phát hiện lỗi, dễ bảo trì.

c) Hình thái

Hình thái (morphology) của lược đồ cũng là một biểu hiện của chất lượng.

Khi đi từ một mức trên xuống mức dưới, thì các nhánh của LCT có thể xòe ra (rẽ thành nhiều nhánh) hay chụm vào (chập vào cùng một môđun). Nhìn một cách toàn thể thì LCT nên xòe ở trên và chụm lại ở dưới. Thông thường thì trong một LCT, các môđun càng ở phía trên thì chức năng của chúng càng nặng về điều khiển và càng nhẹ về xử lý. Trái lại các môđun càng ở thấp phía dưới thì càng ít tính chất điều khiển và càng nặng về xử lý hay dịch vụ. Bởi vậy càng xuất hiện nhiều các điểm chụm ở phía dưới có nghĩa là có nhiều dịch vụ được sử dụng đi sử dụng lại nhiều lần, ở nhiều chỗ. Sử dụng lại là một trong những mục tiêu của thiết kế môđun, thể hiện trong LCT ở sự chụm lại đó.

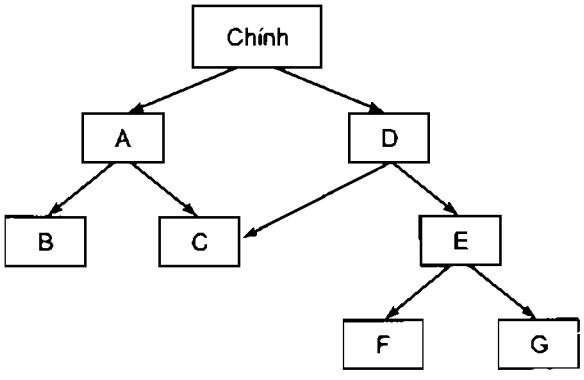
Ta gọi *phạm vi điều khiển* của một môđun là phần LCT bao gồm môđun đó và những môđun phụ thuộc (được gọi) trực tiếp hay gián tiếp từ nó. Ta gọi *phạm vi ảnh hưởng* của một quyết định là phần LCT bao gồm mọi môđun chịu ảnh hưởng của quyết định đó. Chẳng hạn trong LCT cho trong Hình VII.25, thì:

+ phạm vi điều khiển của A là A, B, C;

+ chẳng hạn trong B có một quyết định q1 và kết quả của quyết định này được dùng trong A, E, F, thì phạm vi ảnh hưởng của q1 là A, E, F.

Một thiết kế tốt phải tạo ra LCT, trong đó:

- các quyết định phải có miền ảnh hưởng càng hẹp càng tốt;
- mỗi phạm vi ảnh hưởng phải nằm trong phạm vị điều khiển tương ứng.



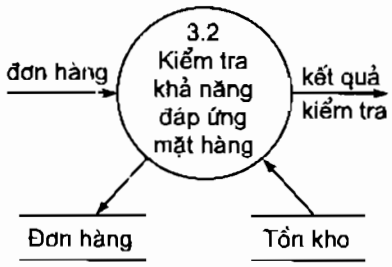
Hình VII.25 Một LCT

5. Đặc tả các môđun

Sau khi lập được LCT (cho mỗi hệ thống con), thì ta còn phải đặc tả mỗi môđun trong đó, tức là miêu tả rõ nội dung của môđun.

Công việc này cũng giống như trước đây, khi ta cần phải đặc tả chức năng xử lý khi đã thành lập BLD (xem §1.5 Chương III). Các phương tiện biểu diễn dùng cho đặc tả môđun vẫn là các phương tiện dùng cho đặc tả chức năng (đặc biệt là các sơ đồ khối hay các ngôn ngữ tựa NNLT). Tuy nhiên điểm khác biệt cơ bản là ở mức độ của sự miêu tả. Đặc tả chức năng của BLD chỉ dừng lại ở mức độ miêu tả lôgic của xử lý, bỏ qua mọi điều kiện cài đặt cụ thể. Còn đặc tả môđun chương trình thì phải cụ thể hơn, bao gồm cả các chi tiết như là: các tham số chuyển giao, các đối thoại với người dùng, các xử lý lỗi, thực hiện vào/ra, tra cứu cơ sở dữ liệu. Vậy ở đây ta phải vận dụng các kết quả thiết kế trước đó, như là thiết kế các thủ tục thủ công, các giao diện, các tệp hay CSDL.

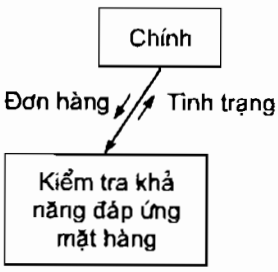
Để thấy rõ hơn, ta hãy xét một thí dụ. Hình VII.26 (a) cho một chức năng trong BLD là "Kiểm tra khả năng đáp ứng một mặt hàng" cùng với đặc tả lôgic của nó.




```
Lấy một đơn hàng
FOR mỗi mặt hàng trong đơn hàng
      BEGIN Đối sánh với lượng hàng tồn kho
        IF không đáp ứng đủ
          THEN BEGIN Báo thiếu hàng
            CASE trả lời
              • Hủy đơn hàng:
                  Báo hủy
              • Rút bớt số lượng:
                  Khấu trừ lượng hàng đặt
              • Bỏ mặt hàng:
                  Loại mặt hàng khỏi đơn hàng
            END
          END
      END
Cất đơn hàng
```

Hình VII.26 (a) Một chức năng và đặc tả

Hình VII.26 (b) cho LCT tương ứng với đặc tả của môđun chương trình thể hiện chức năng 3.2 ở trên.




```
KIỂM TRA KHẢ NĂNG ĐÁP ỨNG MH (đơn hàng, tình trạng)
  FOR mỗi mặt hàng trong đơn hàng
     BEGIN Đọc tệp TỒN KHO, sử dụng SHMH làm khóa
        Tình trạng = 'OK'
        IF Lượng đáp ứng < Lượng yêu cầu
        THEN BEGIN Gửi 'thông báo 1' cho người dùng
                CASE trả lời
                   • Hủy đơn hàng:
                        Thực hiện hủy đơn hàng
                        Tình trạng = 'Hủy'
                   • Rút bớt số lượng:
                        Lượng yêu cầu = Lượng yêu cầu -
                                        Lượng cắt giảm
                        Tình trạng = 'Giảm'
                   • Bỏ mặt hàng:
                        Loại mặt hàng khỏi đơn hàng
                        Tình trạng = 'Cắt MH'
                END
        END
  END
Lưu đơn hàng vào tệp ĐƠNHÀNG
```

Hình VII.26 (b) LCT và đặc tả của môđun tương ứng

Chú thích: Dẫu sao thì đặc tả môđun như trên vẫn chưa là chương trình thực sự, vì ở đây ta chưa dùng một NNLT cụ thể nào cả. Các lệnh trong đặc tả như là "Đọc tệp TỒN KHO" hay "Lưu đơn hàng vào tệp ĐƠNHÀNG " vẫn chỉ là các lệnh macro, khi lập trình còn phải chuyển thành nhiều câu lệnh trong NNLT cụ thể. Tuy nhiên một đặc tả như trên là hoàn toàn sẵn sàng để chuyển sang lập trình được.

6. Đóng gói thành môđun tải

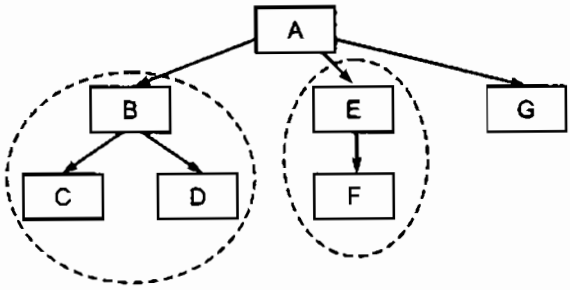
Một môđun chương trình muốn thực hiện được phải đưa vào bộ nhớ trong (gọi là được nạp, hay được tải). Mỗi lần tải phải tốn một phần thời gian của hệ thống.

Người ta gọi môđun tải (load modul) là một nhóm môđun chương trình được tải vào bộ nhớ trong đồng thời. Lý tưởng nhất là tất cả các môđun trong một LCT hợp thành một môđun tải. Như thế thì thời gian tiêu tốn cho việc tải chương trình là ít nhất, và chương trình chạy một cách thuận lợi, vì gọi môđun nào là môđun đó đã sẵn sàng. Tuy nhiên như vậy đòi hỏi bộ nhớ lớn, nhiều khi không đáp ứng nổi. Ngược lại, nếu mỗi môđun chương trình là một môđun tải, thì tiết kiệm được chỗ nhớ, nhưng lại tốn nhiều thời gian. Vì vậy cần tìm giải pháp trung gian cho môđun tải. Bởi vậy nhiệm vụ cuối cùng trong thiết kế chương trình là cắt LCT thành các môđun tải.

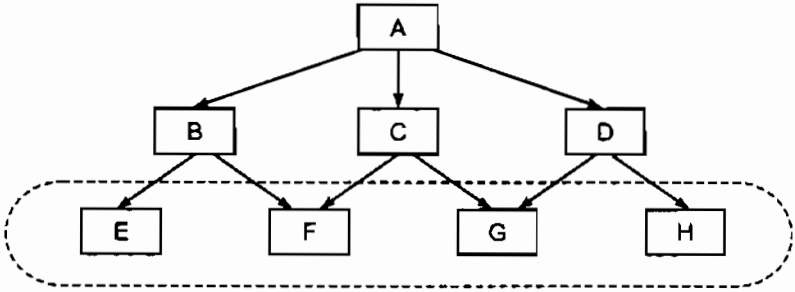
Nói chung thì việc thiết kế môđun tải phải căn cứ trên các yếu tố như là kích cỡ bộ nhớ, kích cỡ các môđun, tần suất các cuộc gọi môđun và một môđun tải phải bao gồm các môđun gắn kết với nhau nhiều nhất.

Tuy nhiên không có một phương pháp rõ rệt cho việc thiết kế môđun tải. Các Hình VII.27 (a), (b), (c) cho một số cách làm, mang tính gợi ý mà thôi.

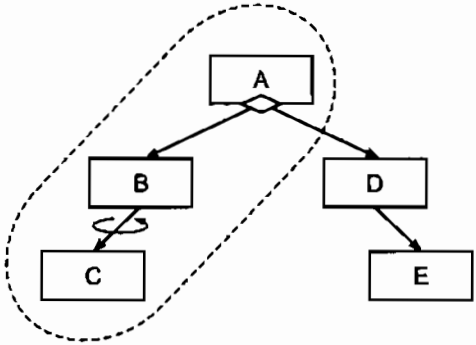
Hình VII.27 (a) gom các môđun theo phạm vi điều khiển (tức là theo dòng các lời gọi). Hình VII.27 (b) lại gom mọi môđun ở mức thấp nhất vào một môđun tải. Đương nhiên cách làm này chỉ có ích khi các môđun ở mức thấp này là những môđun dùng chung và dùng luôn cho các môđun ở những mức trên, chúng cần phải thường trực ở bộ nhớ trong. Hình VII.27 (c) lại đi sâu hơn vào các cấu trúc chọn hay lặp. Ở đây A thường xuyên chọn B và thảng hoặc lắm mới chọn D, mà B lại gọi lặp C nhiều lần. Vậy gom A, B, C vào một môđun tải (nếu bộ nhớ là eo hẹp).



Hình VII.27 (a) Gom môđun tải theo dòng điều khiển



Hình VII.27 (b) Gom môđun tải theo mức thấp nhất



Hình VII.27 (c) Gom môđun tải theo tần số gọi

Chương VIII
MỘT SỐ VẤN ĐỀ VỀ CÀI ĐẶT HỆ THỐNG

Trọng tâm của cuốn sách này là phân tích và thiết kế hệ thống (như đã chỉ rõ trong đề mục của nó). Như vậy các việc còn lại sau phân tích và thiết kế như là cài đặt, khai thác và bảo trì là ra ngoài phạm vi của nó. Tuy nhiên trong chương cuối cùng này, một số vấn đề chọn lọc về cài đặt (lập trình và kiểm định) sẽ được trình bày nhằm vào hai mục đích:

- Giới thiệu cách cài đặt trên xuống, như là một sự tiếp nối tự nhiên và hợp lý của quá trình phân tích và thiết kế trên xuống;
- Giới thiệu một môi trường cài đặt hiện đại, đó là môi trường client/server. Con đường đi từ phương pháp phân tích thiết kế cổ điển sang xu hướng cài đặt hiện đại dường như là không được tự nhiên lắm, song đó lại chính là sự chọn lựa của một số hệ thống lớn như là ORACLE.

§1. CÀI ĐẶT TRÊN XUỐNG VÀ TĂNG TRƯỞNG

Phương pháp phân tích và thiết kế mà ta đã dùng là phương pháp trên xuống. Cho nên một cách tự nhiên, thì phương pháp cài đặt, tức là cách thực hiện việc lập trình và kiểm định, cũng theo xu hướng trên xuống. Để thấy rõ ưu điểm của phương pháp này ta hãy xét phương pháp đối lập với nó là phương pháp cài đặt dưới lên.

1. Cài đặt dưới lên

Đây là cách làm mà không ít người vẫn áp dụng. Việc cài đặt thực hiện theo trật tự sau:

- lập trình tất cả các môđun, lần lượt từ dưới lên;
- kiểm định làm sau khi lập trình hoàn tất và lần lượt:

  + kiểm định môđun,

+ kiểm định tích hợp,

+ kiểm định hệ thống.

Nhược điểm rõ ràng nhất của phương pháp này là: Khi phát hiện (qua kiểm định) một lỗi ở giai đoạn muộn, thì phải xét lại những kết quả tưởng là đã hoàn tất trước đó, có khi gây đổ vỡ tất cả, rất lãng phí công sức.

2. Các nguyên tắc của sự cài đặt trên xuống và tăng trưởng

Có ba nguyên tắc chính chi phối phương pháp này là:

- lập trình và kiểm định đồng thời,
- tăng trưởng dần dần,
- triển khai trên xuống.

Dưới đây ta nói rõ hơn về việc thực hiện ba nguyên tắc này.

a) Lập trình và kiểm định đồng thời

Không như ở phương pháp cài đặt dưới lên là lập trình xong mới kiểm định, ở phương pháp này lập trình xong môđun nào thì thực hiện kiểm định ngay môđun đó, rồi tích hợp luôn môđun đó vào phần hệ thống đã hình thành, cứ thế cho đến khi tích hợp cả hệ thống. Như thế lập trình và kiểm định xen kẽ nhau theo từng môđun.

b) Tăng trưởng dần dần

Đây là một quá trình lặp, diễn ra theo sơ đồ sau:

```
REPEAT
        lập trình một môđun và kiểm định nó
        thêm môđun này vào tổ hợp đã hình thành
        kiểm định và gỡ rối tổ hợp mới
UNTIL hệ thống hoàn thành
bàn giao hệ thống
```

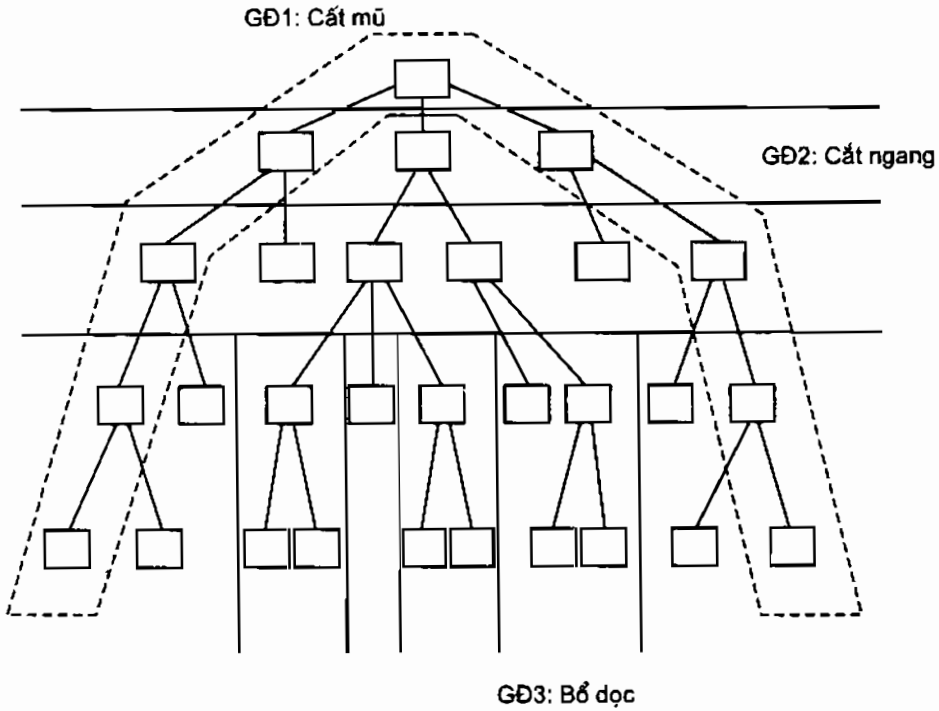
Kết quả của mỗi vòng lặp là một tổ hợp mới hình thành; đây thực chất là một hệ thống chạy được (và do đó kiểm định được), mặc dầu nó chưa làm được gì nhiều. Người ta thường gọi đó là một hệ thống xương (skeleton system). Hệ thống xương tăng trưởng dần dần qua các lần lặp, cho đến khi trở thành hệ thống hoàn chỉnh.

c) Triển khai trên xuống

Quá trình lặp nói trên vét cạn dần dần các môđun trong một LCT để đem ra lập trình. Trật tự chọn môđun kế tiếp để đem ra lập trình là trật tự trên xuống, nghĩa là bắt đầu với môđun đỉnh, ta tiến dần xuống dưới cho đến mức thấp nhất trong LCT.

Nói cụ thể hơn thì quá trình này được phân thành ba giai đoạn (xem Hình VIII.1):

- Giai đoạn 1: Giai đoạn này được gọi là "cắt mũ"; các môđun thuộc một lớp phía trên và bên ngoài của LCT (bao gồm cả hai cánh) được bóc dỡ (lần lượt trên xuống). Nói chung, đây chính là phần thực hiện vào/ra mà chưa có xử lý gì.
- Giai đoạn 2: Giai đoạn này được gọi là "cắt ngang"; các môđun còn lại sau giai đoạn 1 được bóc dỡ dần theo từng mức nằm ngang.
- Giai đoạn 3: Giai đoạn này được gọi là "bổ dọc"; khi đã tới vài mức cuối cùng, thì các môđun được lấy ra theo từng lát dọc.



Hình VIII.1 Các giai đoạn cài đặt trên xuống và tăng trưởng

3. Lập trình với các cuống

Để lập trình một môđun, ta dựa vào đặc tả đã thiết lập trước đây của môđun đó. Đương nhiên trong đó phải có những lời gọi đến các môđun phụ thuộc để phó thác một phần việc nào đó cho mỗi môđun phụ thuộc này. Nhưng những môđun này lại chưa có (chưa lập trình và có khi còn cả chưa thiết kế), thì làm sao khi tích hợp vào hệ thống xương mà hệ thống xương đó lại có thể chạy được? Giải quyết việc này bằng cách thế vào chỗ môđun phụ thuộc (cùng với các môđun phụ thuộc dưới môđun này) bằng một *môđun giả* (dummy modul), còn gọi là một *cuống* (stub). Môđun giả chỉ thực hiện một việc rất thô sơ và giả định nào đó, cho nên thành lập nó rất đơn giản.

Như vậy một hệ thống xương bao gồm các môđun đã lập trình, cùng với một số cuống. Bước tiếp theo, khi tích hợp thêm một môđun mới, thì chính là thay thế môđun mới đã lập trình hoàn chỉnh đó vào chỗ cái cuống tương ứng với nó.

Nhiệm vụ thực hiện của một môđun giả, tùy trường hợp, có thể là như sau:

- Không làm gì cả – cách này không có ích mấy, song dễ cài đặt.
- Hiển thị một thông báo nhằm đánh dấu vệt, chẳng hạn "Chào, tôi là Mai vừa mới được gọi". Cách này giúp ích cho việc gỡ rối (debugging).
- Thực hiện một sự trao đổi với người dùng trên một đầu cuống. Như vậy người dùng lại là một cuống, có thể đưa vào một thông tin nào đó tùy thích.
- Trả về một giá trị hằng, ví dụ 100 VNĐ thay vì việc tính tổng thực sự của một hóa đơn.
- Thực hiện nhiệm vụ của môđun thực, nhưng đã giản lược đi rất nhiều.
- Hiển thị hoặc kiểm định dữ liệu mà nó nhận được khi được gọi. Cách làm này mang lại lợi ích hai đằng: môđun này thực hiện kiểm định phần còn lại của hệ thống xương khi nó là cuống, và nó lại được kiểm định bởi phần còn lại của hệ thống xương khi nó là môđun thật.
- Thực hiện một vòng lặp không làm gì, nhằm mô phỏng một khoảng thời gian tiêu tốn. Cách làm này có ích cho các hệ thống thời gian thực hoặc hệ điều hành, trong đó yếu tố thời gian ảnh hưởng tới sự hoạt động đúng đắn của hệ thống.

4. Các ưu điểm của cách cài đặt trên xuống và tăng trưởng

Phương pháp này có nhiều ưu điểm, mà sau đây là những ưu điểm chính:

- Làm đến đâu chắc đến đấy, tránh hay chí ít là giảm được các đảo lộn lớn, các tình trạng đổ vỡ.
- Cho phép thực hiện nhiều phương án trung gian một cách mềm dẻo để đạt được hệ thống mới.
- Sớm thu được các thông tin phản hồi cho phép điều chỉnh các phương án, nhờ vào việc tiến hành kiểm định sớm và cũng nhờ vào việc người cài đặt và cả người dùng thấy được sự hoạt động của hệ thống sớm, để có thể đưa ra ý kiến thay đổi nếu cần.
- Phần lớn các giao diện được cài đặt và kiểm định sớm, tạo ra ngay được bộ mặt của hệ thống.
- Qua từng bước, người cài đặt và cả người dùng chứng kiến được sự hình thành của hệ thống, khiến họ dễ tin tưởng và phấn chấn tinh thần hơn.
- Nếu thiếu thời gian thì lập trình và kiểm định có thể thực hiện ngay cả khi thiết kế chưa hoàn tất.

Chú thích: Kiểm định (test) mà ta đề cập ở trên, bao gồm kiểm định môđun, kiểm định tích hợp và kiểm định hệ thống gọi chung là kiểm định nội bộ, nghĩa là nó được thực hiện bởi người cài đặt và người thiết kế. Còn một loại kiểm định nữa, thực hiện sau khi hệ thống hoàn tất và bàn giao, đó là kiểm định nghiệm thu, được thực hiện bởi người dùng, người phân tích hệ thống cùng với các người cài đặt. Ta không đi sâu vào kỹ thuật kiểm định ở đây; độc giả muốn tìm hiểu sâu hơn về kiểm định xin tìm đọc ở tài liệu khác, ví dụ [7].

§2 CHỌN MỘT MÔI TRƯỜNG CÀI ĐẶT

1. Một môi trường điển hình

Ngày nay việc chọn lựa một môi trường phần cứng và phần mềm cho việc cài đặt hệ thống có thể thực hiện với nhiều khả năng đa dạng và phong phú.

Đương nhiên việc chọn lựa đó phải xuất phát từ nhu cầu thực sự, tùy kích cỡ của hệ thống và những đặc trưng xử lý của hệ thống.

Với giả thiết là hệ thống ta đang xây dựng là một hệ thống xử lý thông tin quản lý của một xí nghiệp cỡ vừa, với yêu cầu thu thập thông tin ngay tại nguồn, thì môi trường phần cứng và phần mềm điển hình được chọn lựa là như sau:

- Một hệ thống phân tán và mở bao gồm một máy chủ và một số trạm làm việc không đồng nhất, các đầu cuối bị động hay các máy vi tính, liên kết với nhau trên một mạng.
- Một hệ QT CSDL quan hệ, tọa vị trên máy chủ, làm nhiệm vụ quản lý tập trung các dữ liệu, cho phép thực hiện các truy nhập song song, lưu giữ và hồi phục khi có sự cố.
- Một phần mềm truyền thông giữa các trạm làm việc của các người dùng cho phép cập nhật CSDL từ xa, tra cứu nó hoặc khởi động các ứng dụng liên quan.
- Một môi trường lập trình theo sự kiện, cho phép từ các trạm làm việc phát triển các ứng dụng một cách dễ dàng và hữu hiệu.

Các đặc trưng của môi trường điển hình nói trên tất cả đều bao hàm trong khái niệm kiến trúc khách–chủ mà ta sẽ đề cập sau đây.

2. Các kiến trúc khách-chủ

Kiến trúc khách–chủ (client–server architecture*) chỉ là một khái niệm lôgic. Nó được vận dụng vào đâu, thì đó là biểu hiện vật lý của nó mà thôi. Chẳng hạn ta gặp kiến trúc này trong nhiều hệ QT CSDL. Cũng có khi khách và chủ chung nhau trên cùng một trạm làm việc (sử dụng các cơ chế truyền thông cục bộ), và nhiều khi khách và chủ ở trên các trạm khác nhau liên kết thông qua một mạng.

Một xử lý khách–chủ là một quá trình cộng tác giữa hai tác tử, một đằng là một hay nhiều khách và đằng khác là một hay nhiều chủ (kẻ phục vụ).

a) Khách

Một khách là một thực thể (quá trình, chương trình, máy tính,...) có yêu cầu truy nhập vào một dịch vụ hay một tài nguyên. Khách, để hoàn thành nhiệm vụ

*) Thuật ngữ Anh client–server dịch ra tiếng Việt là khách–chủ thì quả thật là không sát nghĩa, vì server là kẻ phục vụ, nói thô thiển một chút là... đầy tớ, chứ không thể coi là chủ được. Tuy nhiên ở đây vẫn được gọi là khách–chủ, bởi từ này được quen dùng.

của mình, phải tương tác với một hay nhiều chủ. Khách cũng đảm nhiệm đối thoại với người dùng. Đối thoại này nói chung là sự tương tác bàn phím/ con chuột, sự thu thập hay nghiệm đúng các thông tin, v.v... Sự tương tác này tạo cho người dùng có cảm giác là chỉ trao đổi thông tin với máy tính mà anh ta đang làm việc.

b) Chủ

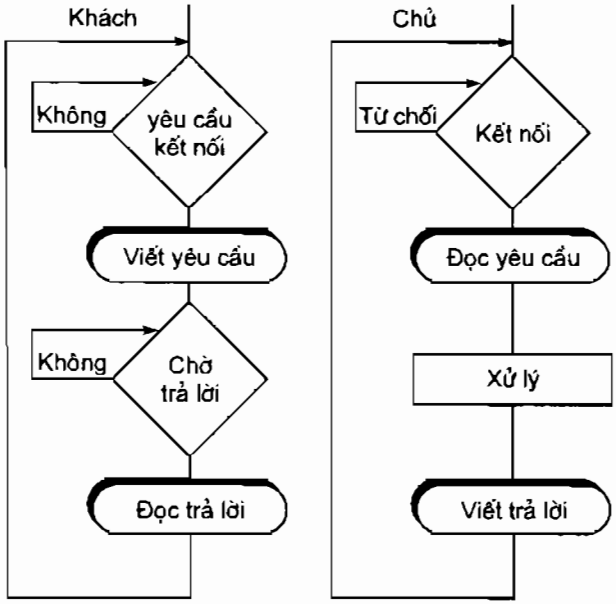
Một chủ là một thực thể (quá trình, chương trình, máy tính,...) trả lời cho một yêu cầu hay cấp phát một tài nguyên. Sau giai đoạn khởi tạo, thì nói chung chủ sẽ chuyển sang giai đoạn chờ đợi một yêu cầu đến từ các khách. Khi chủ nhận được một yêu cầu của khách, nó có thể tự mình xử lý yêu cầu nếu là xử lý rất ngắn, hoặc nếu chỉ có một khách đưa ra yêu cầu (quả vậy, nhiều khách có thể cùng một lúc đưa ra yêu cầu đối với cùng một chủ và nó phải sẵn sàng tiếp nhận). Trong trường hợp ngược lại, chủ có thể viện tới một hay nhiều chủ khác hay quá trình để xử lý yêu cầu của khách. Điều đó cho phép một chủ luôn tập trung vào các yêu cầu tiếp đến của các khách. Sự hoạt động của chủ, mà người ta chỉ định làm chủ chính, luôn luôn trút công việc cho các chủ khác, là một hoạt động quen thuộc trong một số chủ về cơ sở dữ liệu (ORACLE chẳng hạn). Cuối cùng cần lưu ý rằng, trong các mạng tin học, có nhiều loại chủ: chủ tính toán, chủ các tệp, chủ in ấn, chủ hiển thị, v.v...

3. Hoạt động của các hệ thống khách-chủ

Sự liên lạc giữa khách và chủ là thuộc loại giao dịch và hợp tác. Khách chuyển giao một yêu cầu và ngừng việc xử lý của mình cho tới khi nhận được trả lời từ bên chủ. Phương thức hoạt động này của khách gọi là phương thức *phong tỏa*. Còn chủ thì không làm gì cho tới khi nhận một yêu cầu. Ta nói rằng nó ở trạng thái *nghe ngóng*. Khi xử lý xong yêu cầu, chủ lại trở về nghe ngóng yêu cầu khác đến.

Ngoài các hoạt động đơn giản như trên, cũng còn những cơ chế khác, cho phép khách vẫn tiếp tục công việc của mình sau khi gửi một yêu cầu đi. Gọi đó là cách làm việc *không phong toả*. Hình VIII.2 trình bày nguyên tắc liên lạc giữa khách và chủ thông qua cách gửi các thông điệp (messages). Trong trường hợp này thì không cần có một cơ chế tường minh về đồng bộ hóa giữa khách và chủ. Khách viết một yêu cầu của nó vào một vùng đệm, và một khi yêu cầu đó được chủ tiếp nhận thì nó chuyển sang chờ đợi (nếu theo phương thức phong tỏa) hoặc tiếp tục công việc xử lý của mình, khi công việc này không phụ thuộc vào kết

quả trả lời (nếu theo phương thức không phong tỏa). Chủ đọc nội dung của vùng đệm và xử lý yêu cầu (tự nó hay viện đến các chủ khác). Khi xử lý xong, chủ viết



Hình VIII.2 Sự liên lạc giữa khách và chủ

trả lời vào một vùng đệm mà khách truy nhập được (ở phương thức không phong tỏa, thì phải có một cơ chế cho phép khách đọc ngay trả lời khi nó đến, hay đọc muộn lại tùy theo nó có ngắt tạm được công việc đang làm hay không). Chủ lại trở về trạng thái nghe ngóng. Để cơ chế liên lạc này thực hiện được, phải có một giao diện cho phép khách và chủ truyền thông cho nhau. Giao diện này gọi là *giao diện lập trình ứng dụng*, thường gọi dưới cái tên quen thuộc là các API (Application Programming Interface).

Mỗi API có một ngữ pháp riêng. Chẳng hạn trong Unix, các ứng dụng phân tán được xây dựng theo mô hình khách–chủ, thì dùng API socket (lỗ cắm). Giao diện socket gồm một tập hợp các nguyên thủy cho phép truy nhập vào mạng mà không cần quan tâm tới phần vận chuyển thông tin. RPC (Remote Procedure Call) là một loại API khác được thành lập bên trên các socket (đối với RPC của Sun). RPC cho phép thiết kế các ứng dụng phân tán theo mô hình khách chủ với một giao diện ở mức cao hơn các socket. Mục tiêu của API này là tạo khả năng cho một khách có thể yêu cầu một chủ thực hiện một hàm hay một thủ tục trên

máy ở xa. Đương nhiên, API được biết đến nhiều nhất là API truy nhập vào các cơ sở dữ liệu : đó là SQL. Quả vậy, một ứng dụng khách muốn truy nhập vào một CSDL quan hệ, sẽ dịch yêu cầu của nó sang SQL để chuyển tới server của CSDL. Các câu truy vấn sẽ được dịch sang ngôn ngữ thao tác nội bộ, được đem thực hiện và gửi lại trả lời cho khách. Trả lời thường là ở dạng một bảng, hoặc một trang hay đối tượng. Khi một ứng dụng Windows cần truy nhập từ xa vào một hay nhiều servers của CSDL, thì có thể sử dụng API ODBC (Open Database Connectivity) của Microsoft. ODBC được thiết lập xung quanh hai thành phần quan trọng:

- Một thư viện các hàm, được các ứng dụng sử dụng để kết nối, gửi đi và nhận lại các dữ liệu và cắt kết nối.
- Một hệ quản lý các drivers cho phép nạp hệ quản lý CSDL (driver) thích hợp. Mỗi driver, thường do nhà cung cấp hệ QT CSDL thiết lập, bảo đảm sự trao đổi giữa các ứng dụng và CSDL, đặc biệt là dịch ngữ pháp SQL của ODBC sang một ngôn ngữ khác thích hợp với server mà ta hướng tới.

TÀI LIỆU THAM KHẢO

[1] Bénassy J., Les logiciels de gestion, Hermès, Paris 1992.

[2] Carlier A., Le développement du logiciel, Hermès, Paris 1995.

[3] Cornes R., System Analysis for Profitable Business Applications, Prentice Hall, New York 1989.

[4] De Marco T., Structured Analysis and System Specification, Yourdon Press, New York 1979.

[5] Foucaut O., Thiery O., Conception des systèmes d'Information et Programmation événementielle, InterEditions, Paris 1996.

[6] Hawryczkiewycz I.T., Introduction to Systems Analysis and Design, Prentice Hall of India, New Delhi 1989.

[7] Jorgensen P.C., Software Testing, CRC Press, New York 1995.

[8] Lavret P., Analyse des systèmes: de l'approche fonctionnelle à l'approche objet, InterEditions, Paris 1994.

[9] Licker P.S., Fundamentals of Systems Analysis with Application Design, Boyd& Freaser pub.com., Boston 1987

[10] Pages–Jones M., The Practical Guide to Structured System Design, Yourdon Press, New York 1980.

[11] Smart C.,Sims R., Phân tích, Thiết kế, Cài đặt Hệ thống Thông tin Quản lý, Viện Tin học, Hà nội 1990.

[12] Weinberg V., Structured Analysis, Yourdon Press, New York 1978.

[13] Yourdon E., Managing the System Life Cycle, Yourdon Press, New York 1982.

MỤC LỤC

CHƯƠNG VI PHÂN TÍCH HỆ THỐNG VỀ ĐỘNG THÁI

CHƯƠNG VII THIẾT KẾ HỆ THỐNG

TỦ SÁCH ĐÀO TẠO KỸ SƯ TIN HỌC HỆ DÀI HẠN
KHOA CÔNG NGHỆ THÔNG TIN ĐẠI HỌC BÁCH KHOA HÀ NỘI

NGUYỄN VĂN BA

Phân tích và thiết kế hệ thống thông tin

Phân tích và thiết kế (PT&TK) là những phần việc cực kỳ quan trọng trong quá trình xây dựng một hệ thống tin học.

Cuốn sách này trình bày một loạt các phương pháp PT&TK kinh điển: Phương pháp SA dùng cho phân tích chức năng, phương pháp E/A và mô hình quan hệ dùng cho phân tích dữ liệu, phương pháp SART dùng cho phân tích động thái và phương pháp SD dùng cho thiết kế hệ thống. Chúng sẽ được vận dụng bổ sung cho nhau, vì mỗi phương pháp đó chỉ đề cập một phương diện của quá trình PT&TK. Mặt khác chúng là hoàn toàn thích ứng được với nhau vì chúng nhất quán trong xu hướng phân tích trên xuống. Các phương pháp này là dễ hiểu, dễ dùng, hơn nữa đã được thử thách qua thời gian, và cho đến nay vẫn được vận dụng trong một số hệ thống lớn và quen thuộc như ORACLE.

Như vậy cuốn sách là có ích cho các sinh viên ngành công nghệ thông tin và những người bước đầu làm quen với PT&TK. Tuy nhiên nó cũng có thể là tài liệu trợ giúp tốt cho các người xây dựng hệ thống chuyên nghiệp, các kỹ sư phân tích và thiết kế.

VÀI NÉT VỀ TÁC GIẢ

Tốt nghiệp ĐHBK Hà Nội năm 1962 và ở lại trường dạy Toán. Nhận học vị Tiến sĩ năm 1971 ở Ba Lan và giảng dạy tin học ở ĐHBK Hà Nội từ đó cho đến nay. Nhận học hàm Phó Giáo sư năm 1980. Các hướng chuyên môn quan tâm là: Lý thuyết ngôn ngữ và tính toán, Chương trình dịch, Lý thuyết chương trình, Phương pháp luận lập trình, Phân tích và thiết kế hệ thống.

GIÁ: 30.000Đ