

Kỹ nghệ phần mềm

Software Engineering

Đại học Kinh doanh và Công nghệ Hà Nội

Khoa CNTT

GV: Đào Thị Phương

Email: phuongdt102@gmail.com

Page fb: [facebook.com/it.hubt](https://www.facebook.com/it.hubt)

Phone: 0946.866.817

Bài 6: Các hoạt động thiết kế



Nội dung

- Thiết kế kiến trúc
- Thiết kế giao diện

TÀI LIỆU THAM KHẢO



1. Nguyễn Văn Vy, Nguyễn Việt Hà. *Giáo trình kỹ nghệ phần mềm*. Nhà xuất bản Đại học Quốc gia Hà nội, 2008
2. Grady Booch, James Rumbaugh, Ivar Jacobson. *The Unified Modeling language User Guid*. Addison-Wesley, 1998.
3. M. Ould. *Managing Software Quality and Business Risk*, John Wiley and Sons, 1999.
4. Roger S.Pressman, *Software Engineering, a Practitioner's Approach*. Fifth Edition, McGraw Hill, 2001.
5. Ian Sommerville, *Software Engineering*. Sixth Edition, Addison-Wasley, 2001.
6. Nguyễn Văn Vy. *Phân tích thiết kế hệ thống thông tin hiện đại. Hướng cấu trúc và hướng đối tượng*, NXB Thống kê, 2002, Hà Nội.

Thiết kế kiến trúc phần mềm

software achitecture design



Khái niệm kiến trúc

- Kiến trúc phần mềm chỉ **cấu trúc tổng thể của 1 phần mềm và cách thức tổ chức** qua đó cho ta 1 sự tích hợp về mặt khái niệm của 1 hệ thống [SHA95a]
- Thông thường: **thể hiện bằng một biểu đồ phân cấp của các thành phần và quan hệ giữa chúng**
- Đầy đủ: **thể hiện cấu trúc hệ thống theo nhiều góc nhìn khác nhau: tĩnh, động, dữ liệu, triển khai**

[SHA95a] Shaw, M and D. Garlan, Formulation and formalisms in software architecture, volume 100-lecture Notes in computer Science, Springer-verlag, 1995

Vai trò kiến trúc phần mềm



- Không phải là mô hình hoạt động
- là mô hình phân hoạch theo những cách nhìn khác nhau (chức năng, dữ liệu, tiến trình, tĩnh hay động..)
- giúp kỹ sư hệ thống:
 - Phân tích tính hiệu quả của thiết kế đáp ứng được yêu cầu của phần mềm
 - Tìm các giải pháp thay thế cấu trúc ở giai đoạn sớm
 - Giảm các rủi ro liên quan tới cấu trúc

Khái niệm thiết kế kiến trúc



- **Quá trình xác định các hệ con lập thành hệ thống và khung làm việc** để điều khiển & giao tiếp giữa các hệ con với nhau
- **Bắt đầu sớm** ngay từ giai đoạn đầu của thiết kế hệ thống, tiến hành cùng với một số hoạt động đặc tả
- Nó bao gồm việc xác các thành phần chính của hệ thống sự truyền thống giữa chúng

Các bước thiết kế kiến trúc



1. Cấu trúc hóa hệ thống: phân chia hệ thống thành các hệ con (sub-system) độc lập và xác định trao đổi thông tin giữa các hệ con □ xác định các giao diện của chúng
2. Mô hình hóa điều khiển: xác lập mô hình điều khiển giữa các phần khác nhau của hệ thống đã được xác định
3. Phân rã thành các module: phân rã các hệ con thành các module.

- ❖ **Hệ con:** phần hệ thống hoạt động độc lập với các dịch vụ mà các hệ con khác cung cấp
- ❖ **Môđun:** phần hệ thống cung cấp dịch vụ và tương tác cùng phần khác để tạo ra dịch vụ hay sản phẩm

Các mô hình kiến trúc



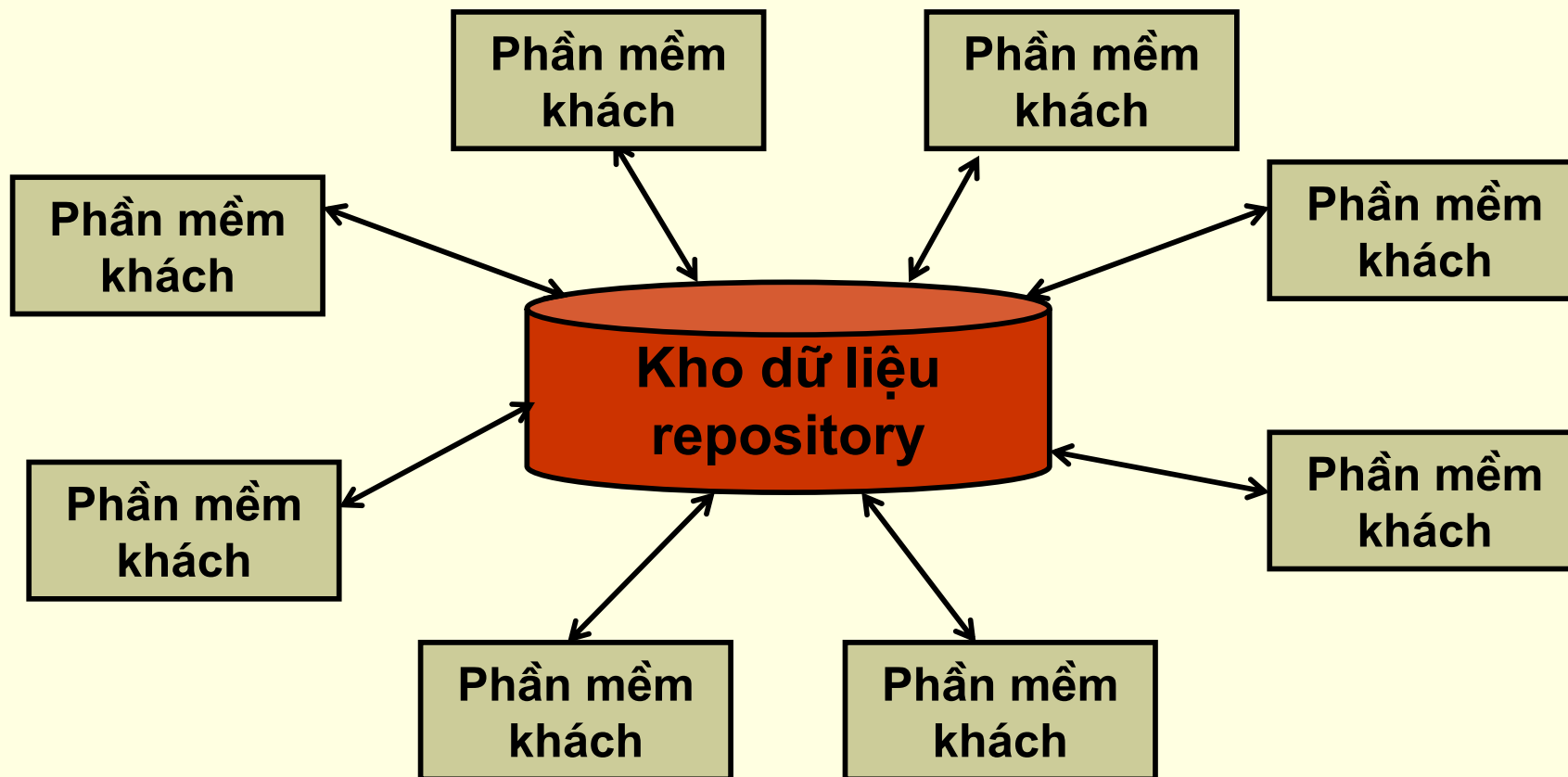
- Các mô hình kiến trúc khác nhau được tạo ra trong quá trình thiết kế
- Mỗi mô hình biểu diễn một cách nhìn của kiến trúc
 - ◆ Mô hình kiến trúc tĩnh chỉ ra các thành phần chính của hệ thống (biểu đồ phân rã)
 - ◆ Mô hình động chỉ ra cấu trúc tiến trình của hệ thống (biểu đồ luồng dữ liệu)
 - ◆ Mô hình giao diện xác định hệ thống giao diện của hệ thống (hệ thống giao diện tương tác)
 - ◆ Mô hình mối quan hệ như mô hình khái niệm thực thể miền dữ liệu của hệ thống

Một số mô hình kiến trúc

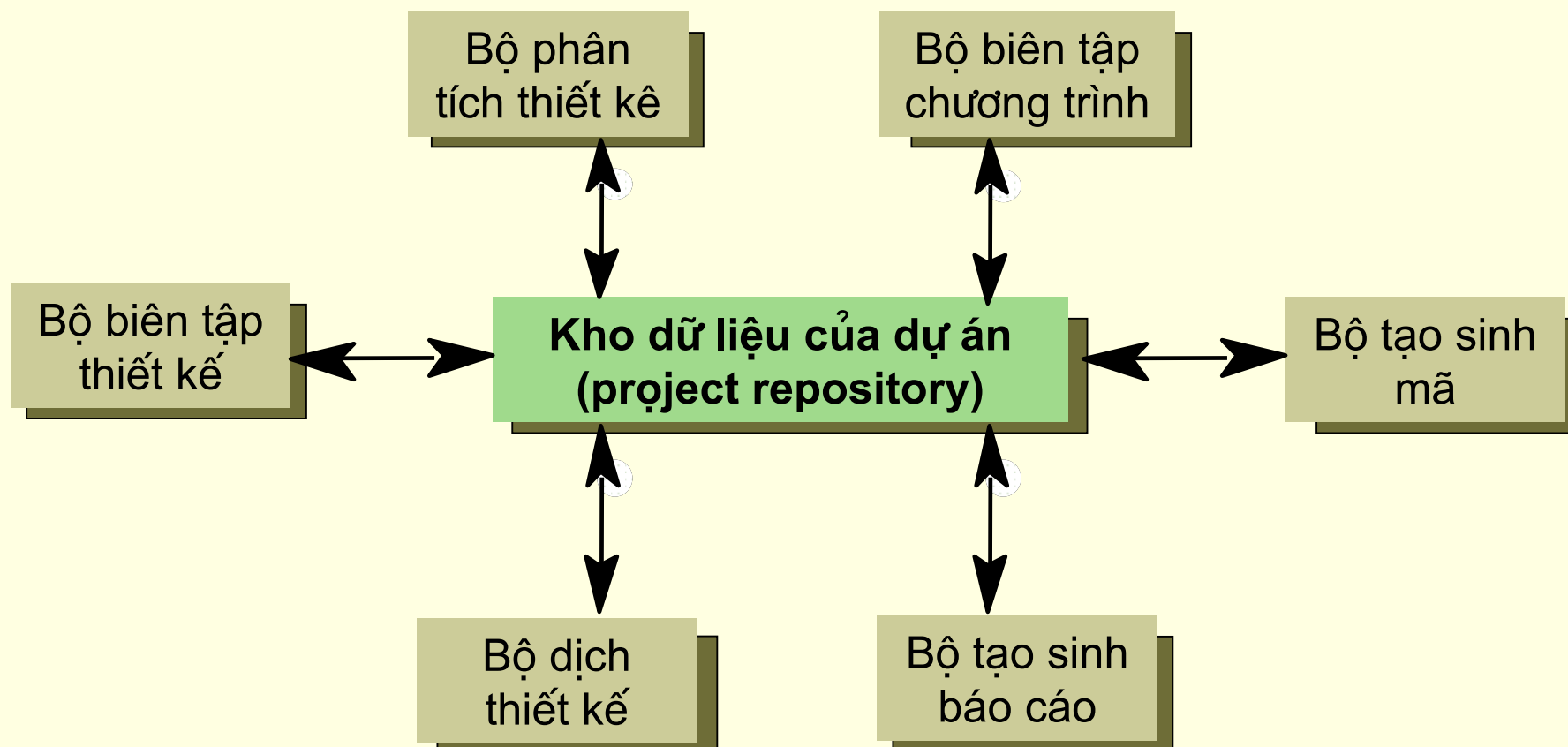


1. Kiến trúc dữ liệu tập trung (**Data-centered architectures**)
2. Kiến trúc khách/dịch vụ (**Client-server architectures**)
3. Kiến trúc phân tầng (**Layered architectures**)
4. Kiến trúc gọi và trả lại (**Call and return architectures**)
5. Kiến trúc luồng dữ liệu (**Data flow architectures**)
6. Kiến trúc hướng đối tượng (**Object-oriented architectures**)

Kiến trúc dữ liệu tập trung



Kiến trúc của bộ công cụ CASE



Kiến trúc dữ liệu trung tâm(t)



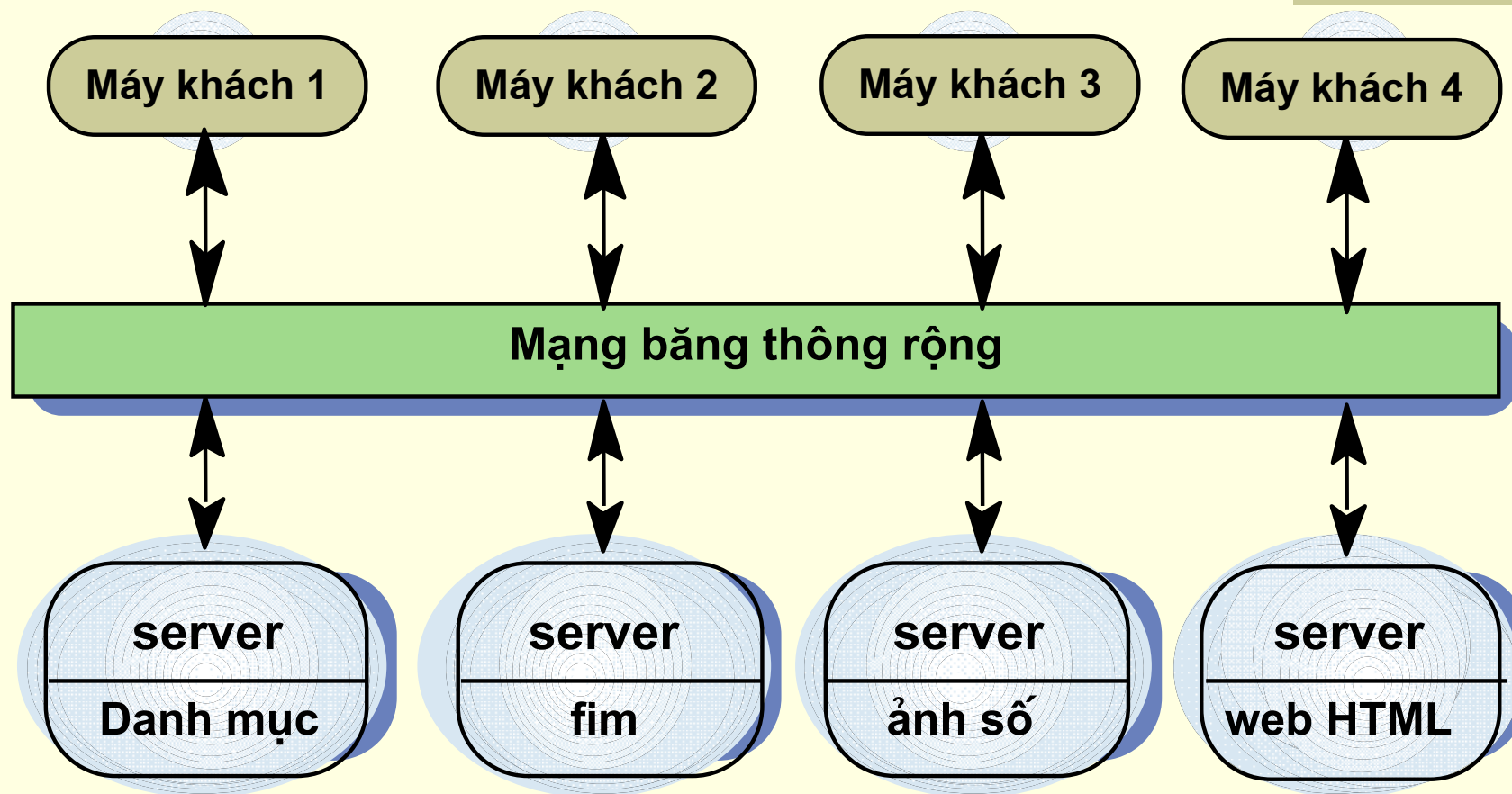
■ Ưu điểm

- Tiện lợi cho chia sẻ dữ liệu lớn
- Phân hệ không cần biết dữ liệu được quản lý và tạo ra như thế nào (sao lưu, bảo mật,..)

■ Nhược điểm

- Các hệ con phải theo mô hình dữ liệu của kho
- Việc tiến hoá dữ liệu là khó khăn và đắt đỏ
- Khó có chính sách quản lý riêng cho các hệ con
- Khó phân bố dữ liệu một cách hiệu quả

Kiến trúc client-server



Kiến trúc client-server (t)



■ Ưu điểm

- ◆ Phân phối dữ liệu trực tiếp
- ◆ Sử dụng hiệu quả mạng, dùng thiết bị rẻ hơn
- ◆ Dễ dàng mở rộng, thêm dịch vụ

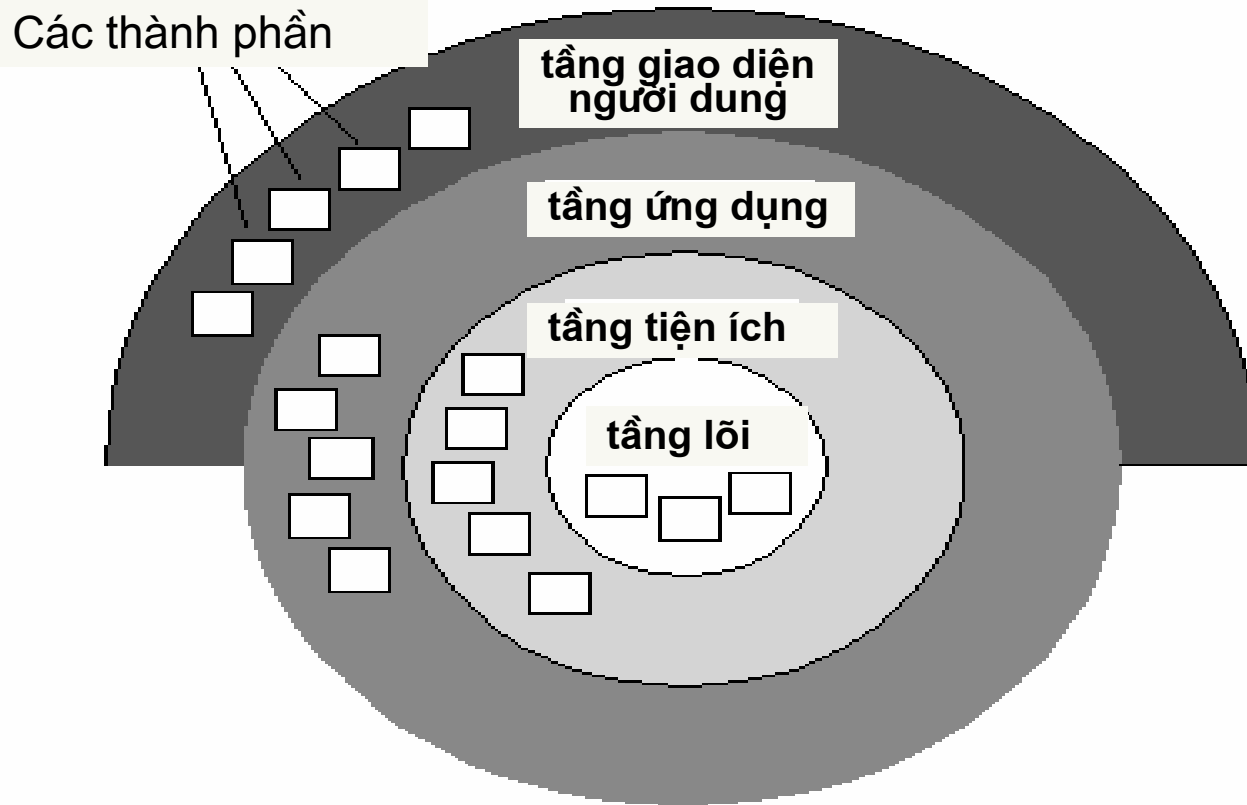
■ Nhược điểm

- ◆ Các hệ con dùng cấu trúc dữ liệu khác nhau không chia sẻ được, trao đổi dữ liệu có thể không hiệu quả
- ◆ Quản lý ở mỗi server là dư thừa
- ◆ Không lưu giữ chung tên và dịch vụ -> khó tìm server hay dịch vụ rồi

Đang là mô hình phát triển ứng dụng phổ biến

Kiến trúc phân tầng

Layered Architecture



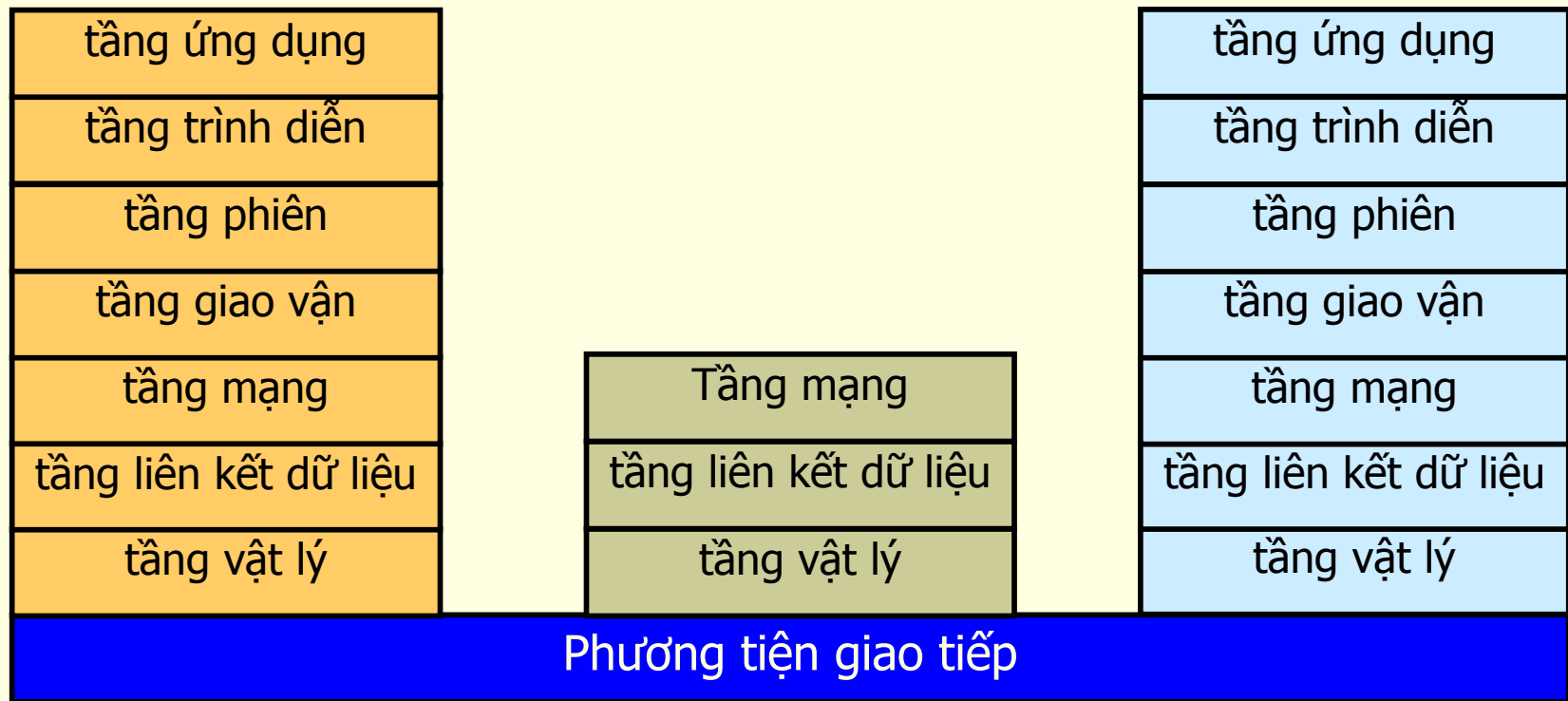
Mô hình máy trừu tượng

Kiến trúc phân tầng (t)



- Dùng để mô hình hóa giao diện của các phân hệ (sub-systems)
- Phân rã hệ thống thành các tầng, mỗi tầng là một tập các dịch vụ
- Hỗ trợ sự phát triển tăng trưởng của các tầng, khi giao diện mỗi tầng thay đổi thì chỉ ảnh hưởng tới các tầng liền kề
- Không phải hệ thống nào cũng dễ dàng phân chia theo mô hình này

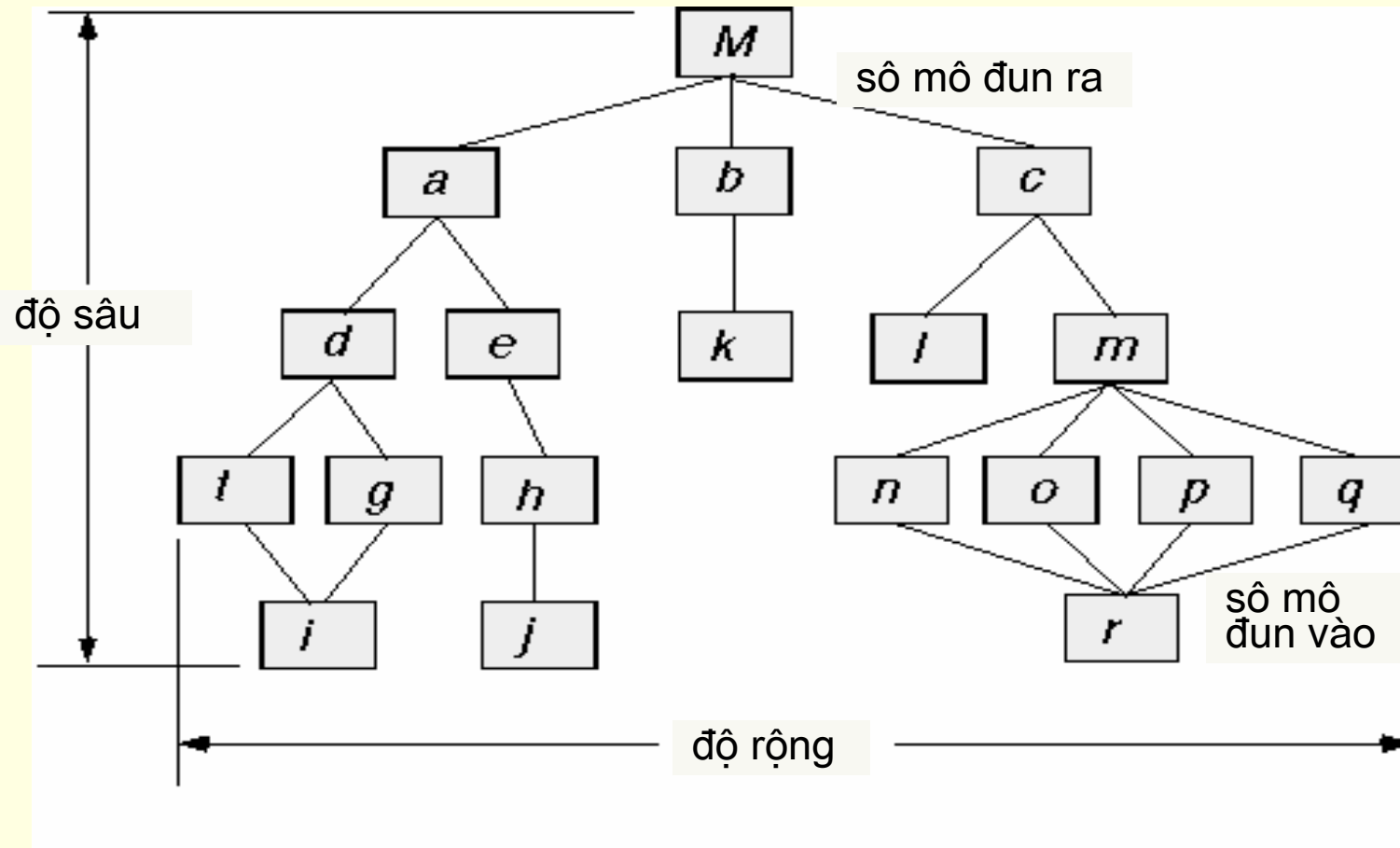
Kiến trúc phân tầng tham khảo OSI



Hình 3.15 Kiến trúc của mô hình tham chiếu OSI

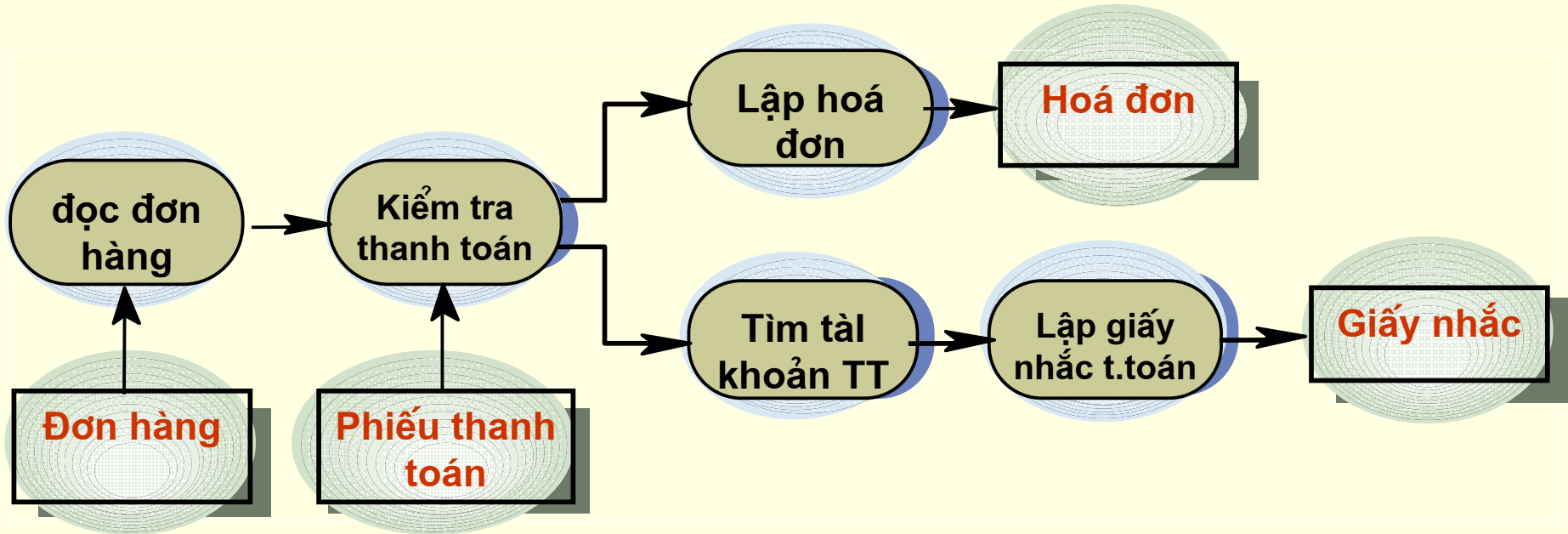
Kiến trúc gọi & trả lại

Call and Return Architecture



Kiến trúc luồng dữ liệu

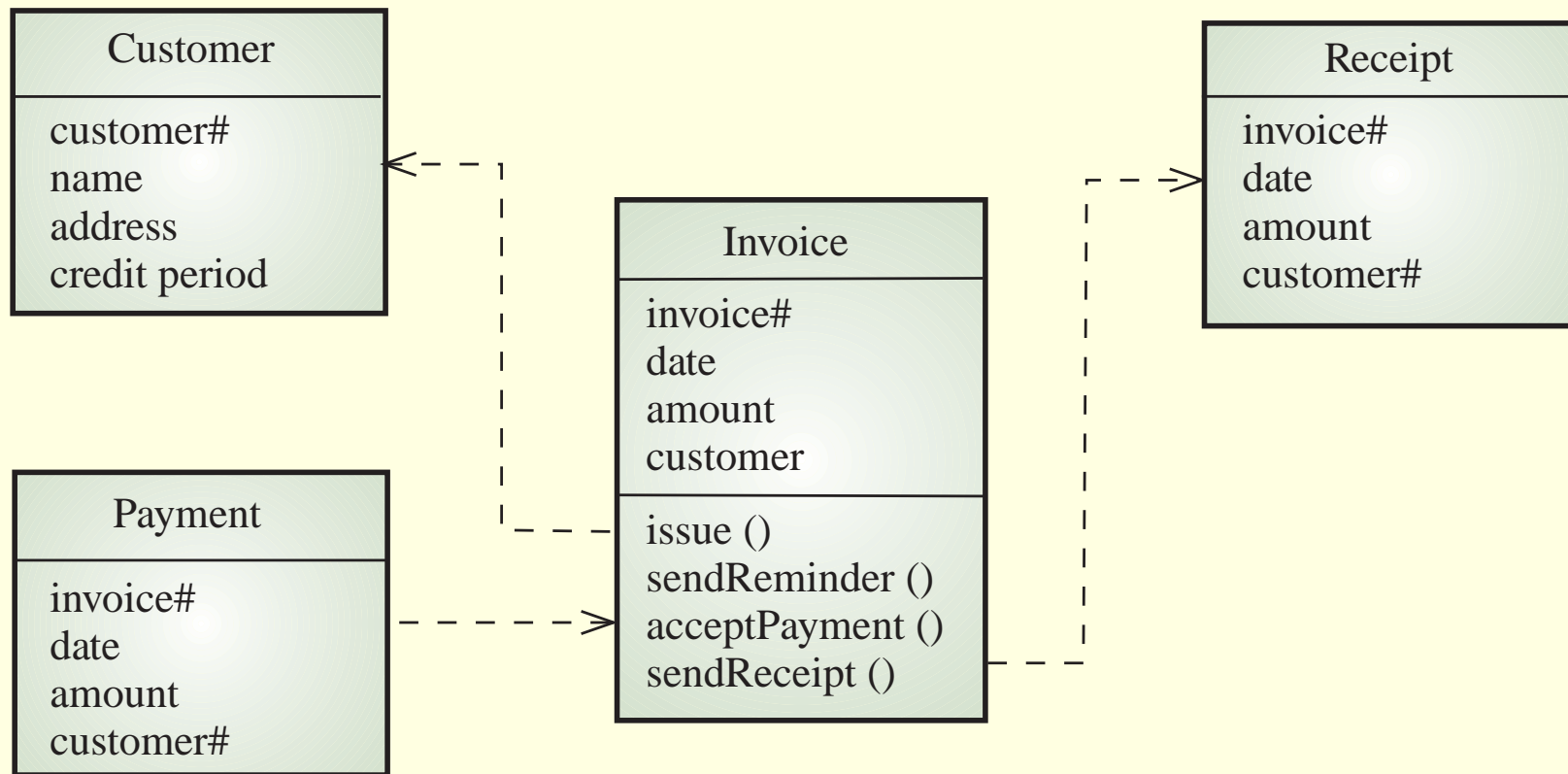
Data Flow Architecture



Hệ thống xử lý đơn hàng

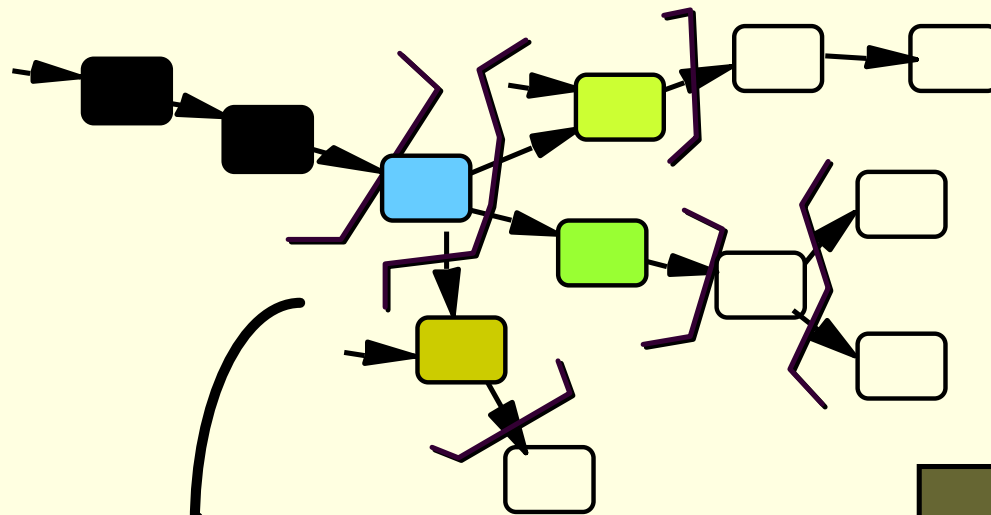
Kiến trúc hướng đối tượng

Object-oriented architecture

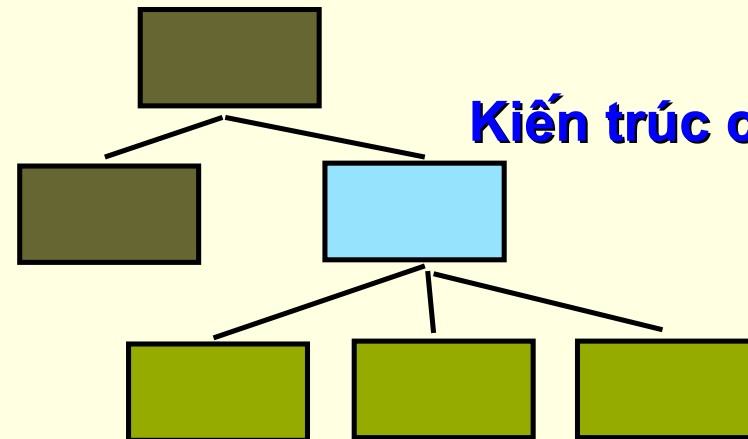


Hệ thống xử lý đơn hàng

Xây dựng kiến trúc chương trình



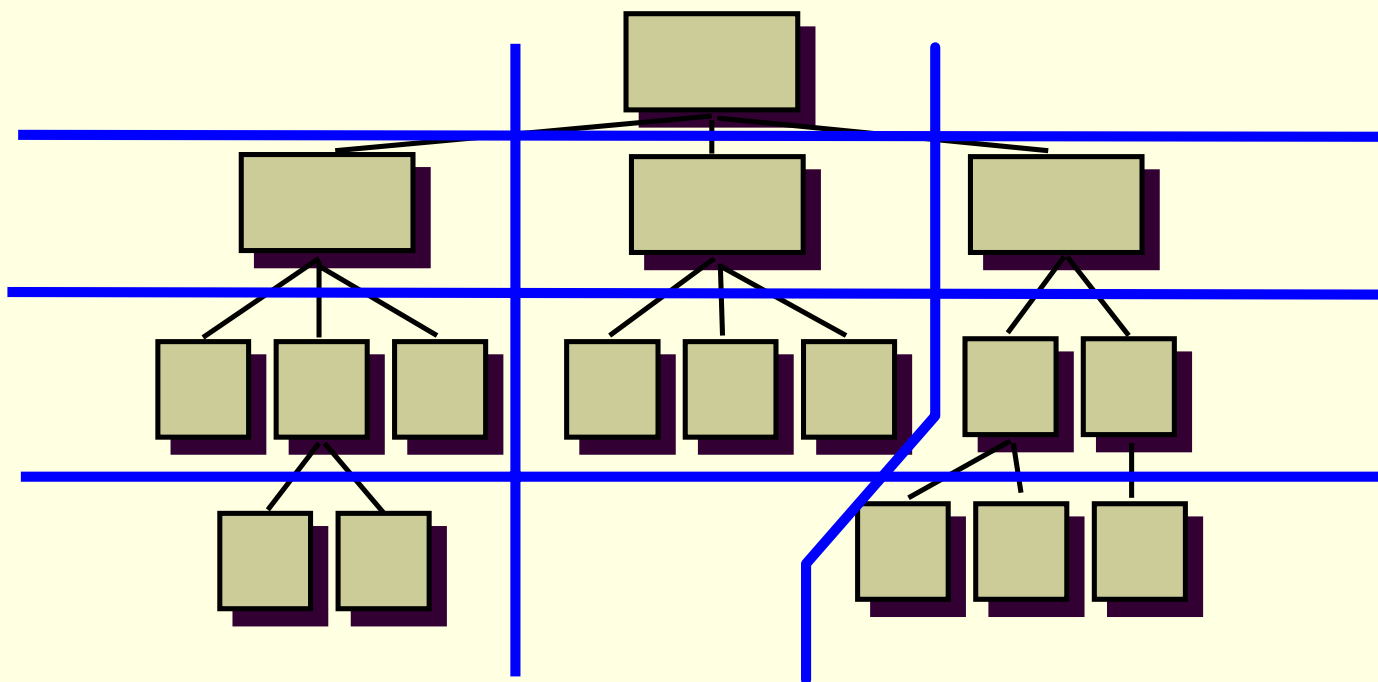
Biểu đồ luồng dữ liệu



Kiến trúc chương trình

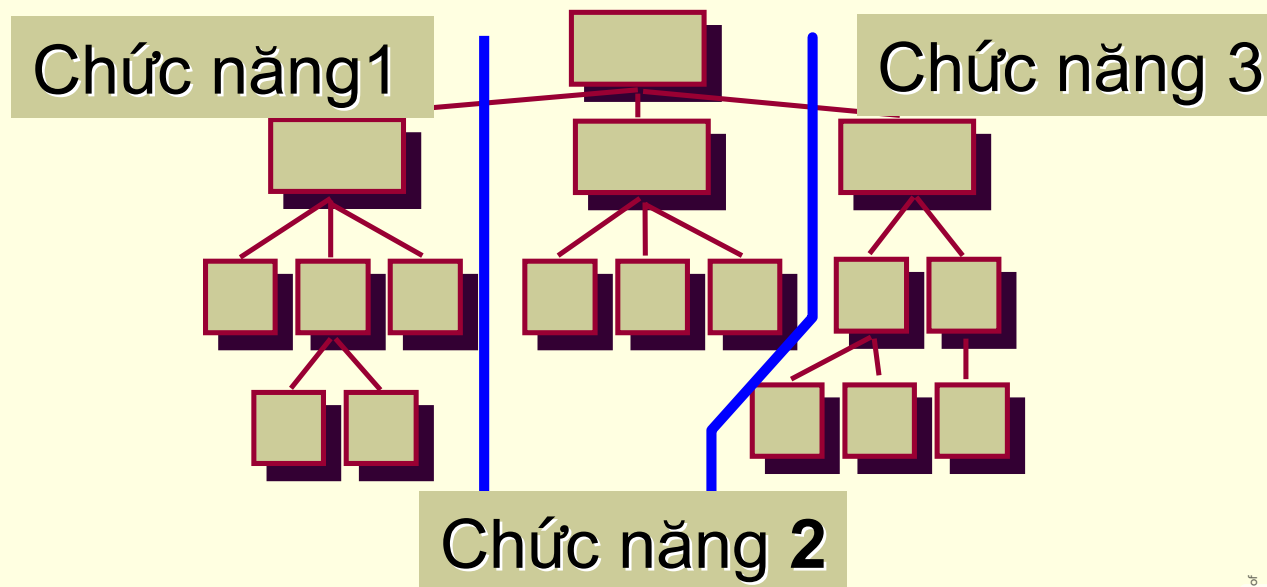
Phân hoạch kiến trúc

- Cần phân hoạch kiến trúc theo chiều: ngang và dọc



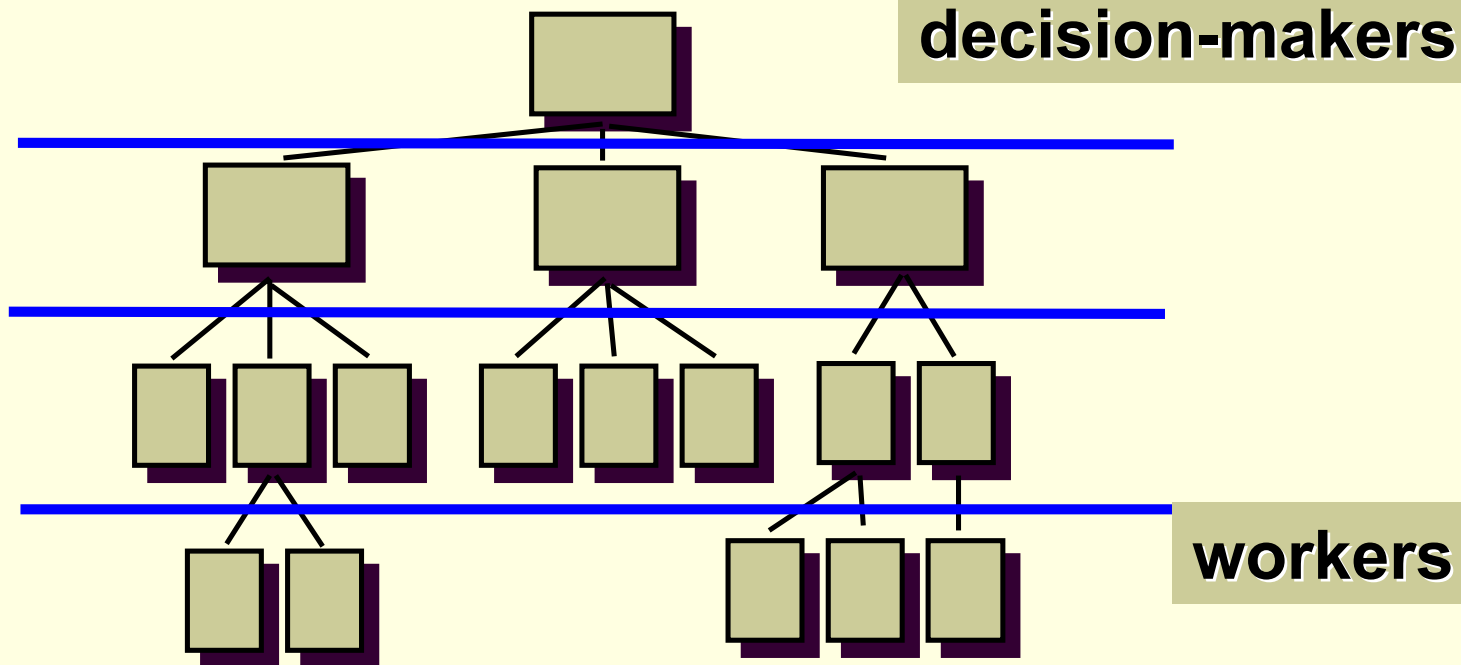
Phân hoạch kiến trúc dọc

- Xác định các nhánh rẽ riêng cho các chức năng chủ chốt
- Sử dụng các module điều khiển để điều phối thông tin giữa các chức năng



Phân hoạch kiến trúc ngang

- phân tầng các module thành từng mức: ra quyết định (điều khiển) và module thao tác
- các module ra quyết định cần được xếp ở tầng cao



Tại sao cần phân hoạch kiến trúc



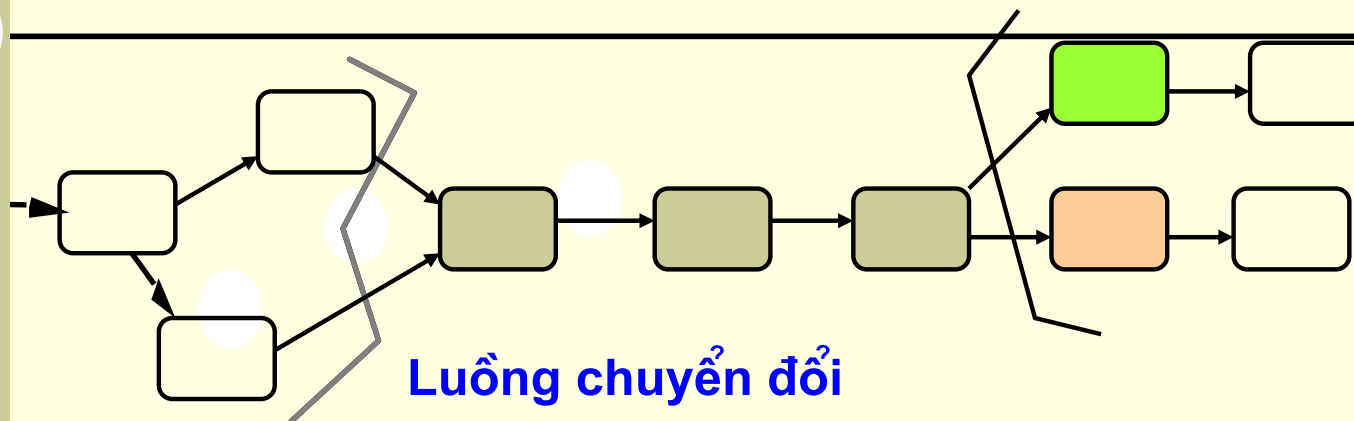
- Tạo ra phần mềm
 - dễ kiểm thử
 - dễ bảo trì
 - hạn chế hiệu ứng phụ khi sửa đổi
 - dễ mở rộng

Thiết kế cấu trúc chương trình



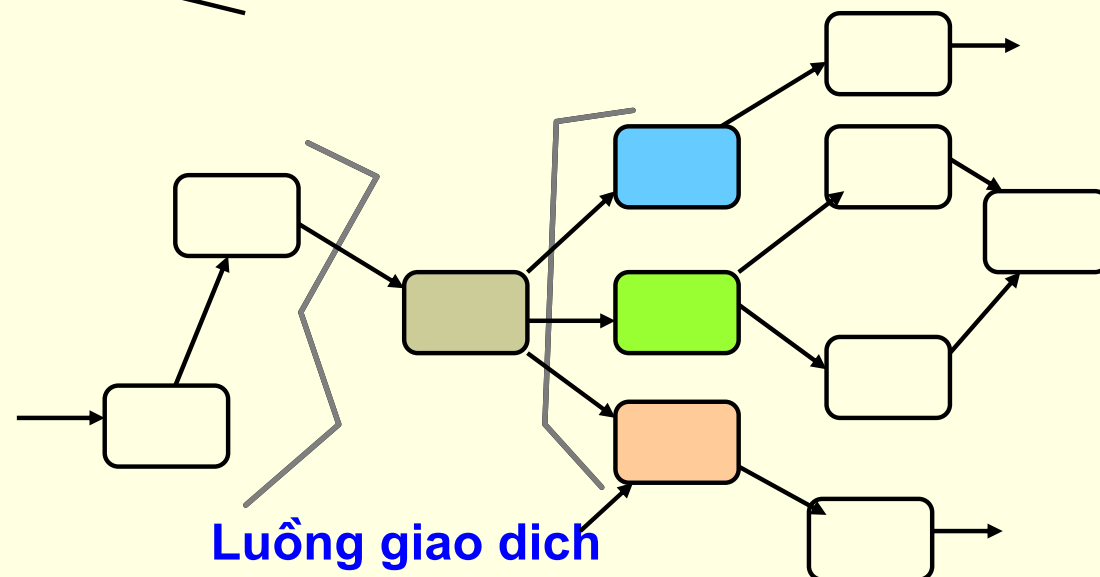
- Mục tiêu: tạo ra module có kiến trúc tốt: được phân hoạch hợp lý, liên kết qua điều khiển
- Cách tiếp cận:
 - chuyển đổi (mapping) DFD thành kiến trúc phần mềm
- Ký pháp: biểu đồ có cấu trúc (structure chart)

Đặc tính của luồng dữ liệu



2 loại luồng dữ liệu tiêu biểu:

- **Luồng chuyển đổi:** xử lý tập trung
- **Luồng giao dịch:** định tuyến phân phối

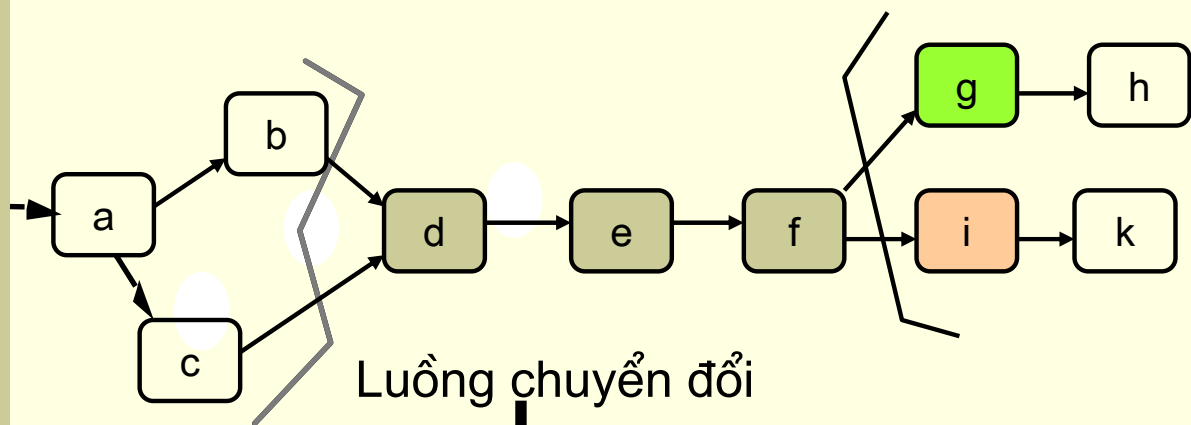


Kiến trúc luồng chuyển đổi

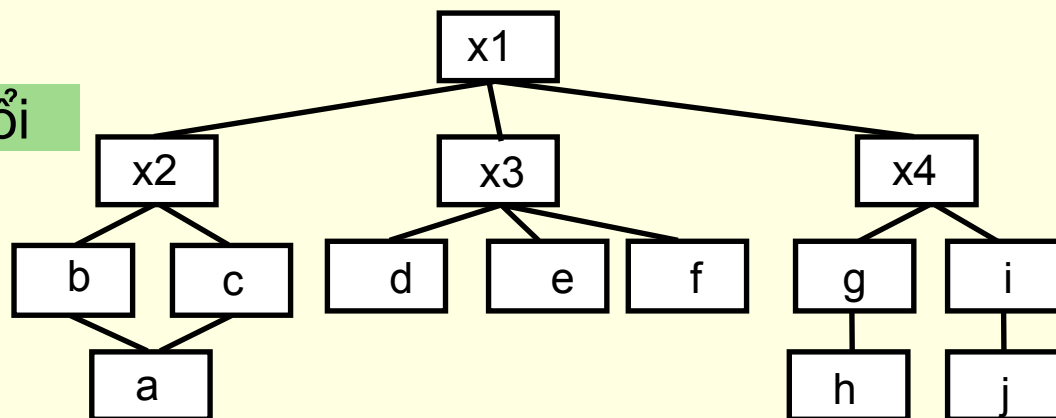


- Cô lập, xác định biên của các module vào/ra; xác định các module **xử lý tập trung**
- Chuyển chúng thành các module kiến trúc tương ứng
- Thêm các module điều khiển nếu cần thiết
- Vi chỉnh (refining) kiến trúc để nâng cao *tính module*
- ❖ Luồng chuyển đổi tập trung là luồng mà có các dữ liệu đầu vào **được tập trung xử lý ở một số tiến trình** rồi cho kết quả đầu ra là các tiến trình thực hiện lưu trữ, truyền đi hay biểu diễn thông tin

Ví dụ: kiến trúc luồng chuyển đổi



ánh xạ luồng chuyển đổi

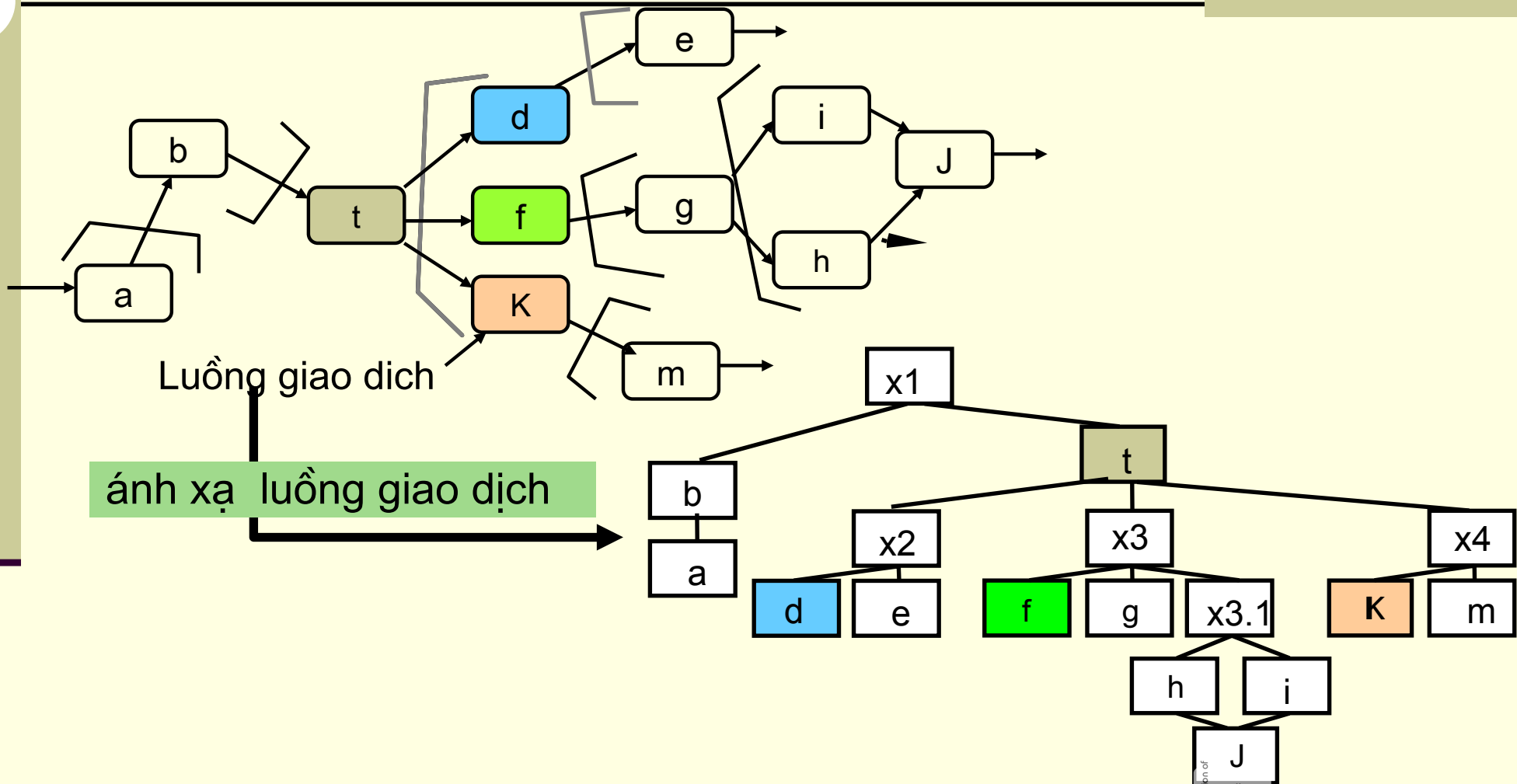


Kiến trúc luồng giao dịch



- Xác định các luồng vào
 - Xác định các luồng thực hiện
 - Xác định **trung tâm giao dịch** (module phân phối)
 - Biến đổi riêng rẽ từng luồng hành động
- ❖ Luồng giao dịch là luồng mà **mỗi đầu vào được nhận dạng và chuyển** cho các tiến trình xử lý tương ứng với nó. Tiến trình làm nhiệm vụ nhận dạng và chuyển dữ liệu đến nơi cần gọi là **trung tâm giao dịch**

Ví dụ: kiến trúc luồng giao dịch



Thiết kế giao diện



Vai trò, tầm quan trọng

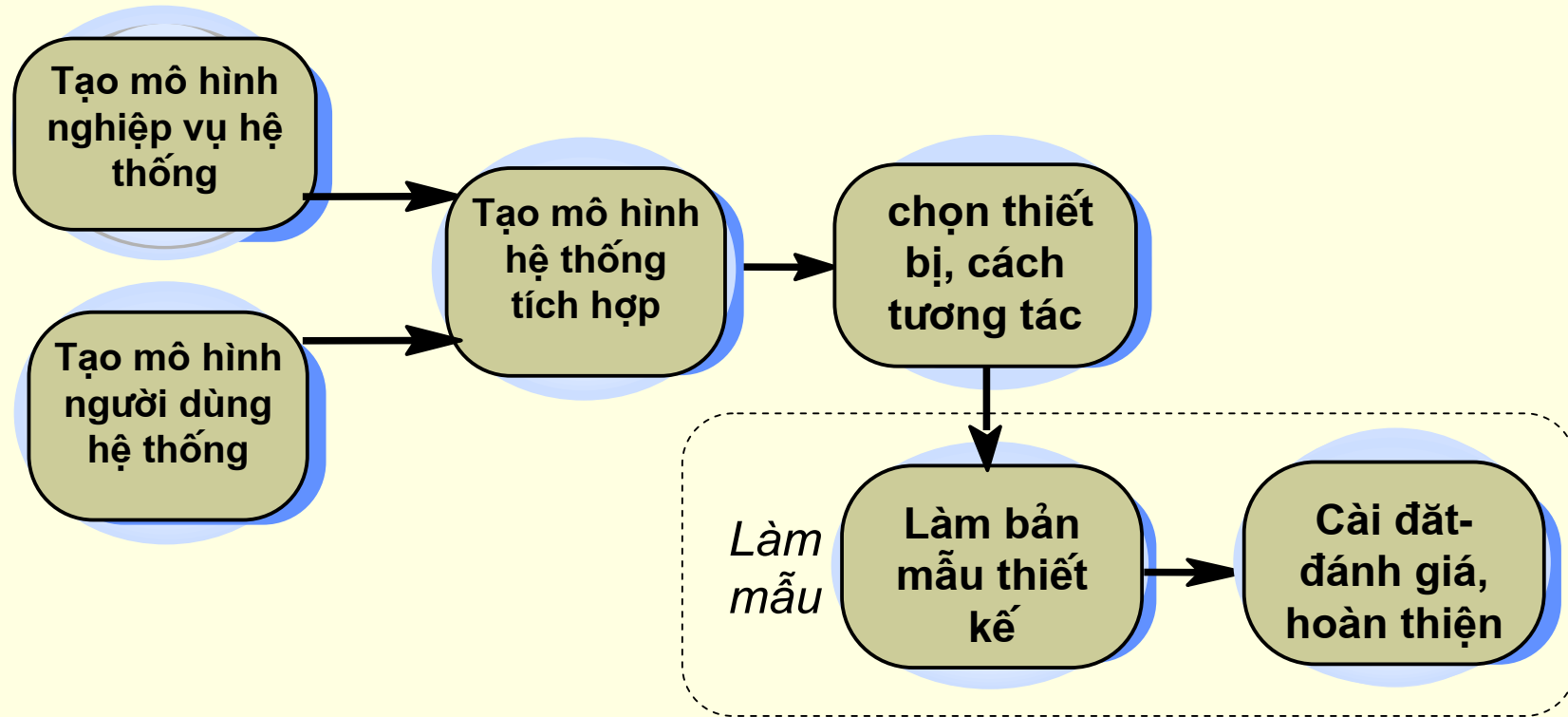
- một khâu không thể thiếu trong thiết kế phần mềm □
người dùng đánh giá phần mềm qua giao diện
- Thiết kế giao diện:
 - ◆ hướng người dùng
 - ◆ che dấu chi tiết kỹ thuật bên trong
 - ◆ kết hợp 3 mắt: người dùng, chức năng, công nghệ

Vai trò, tầm quan trọng

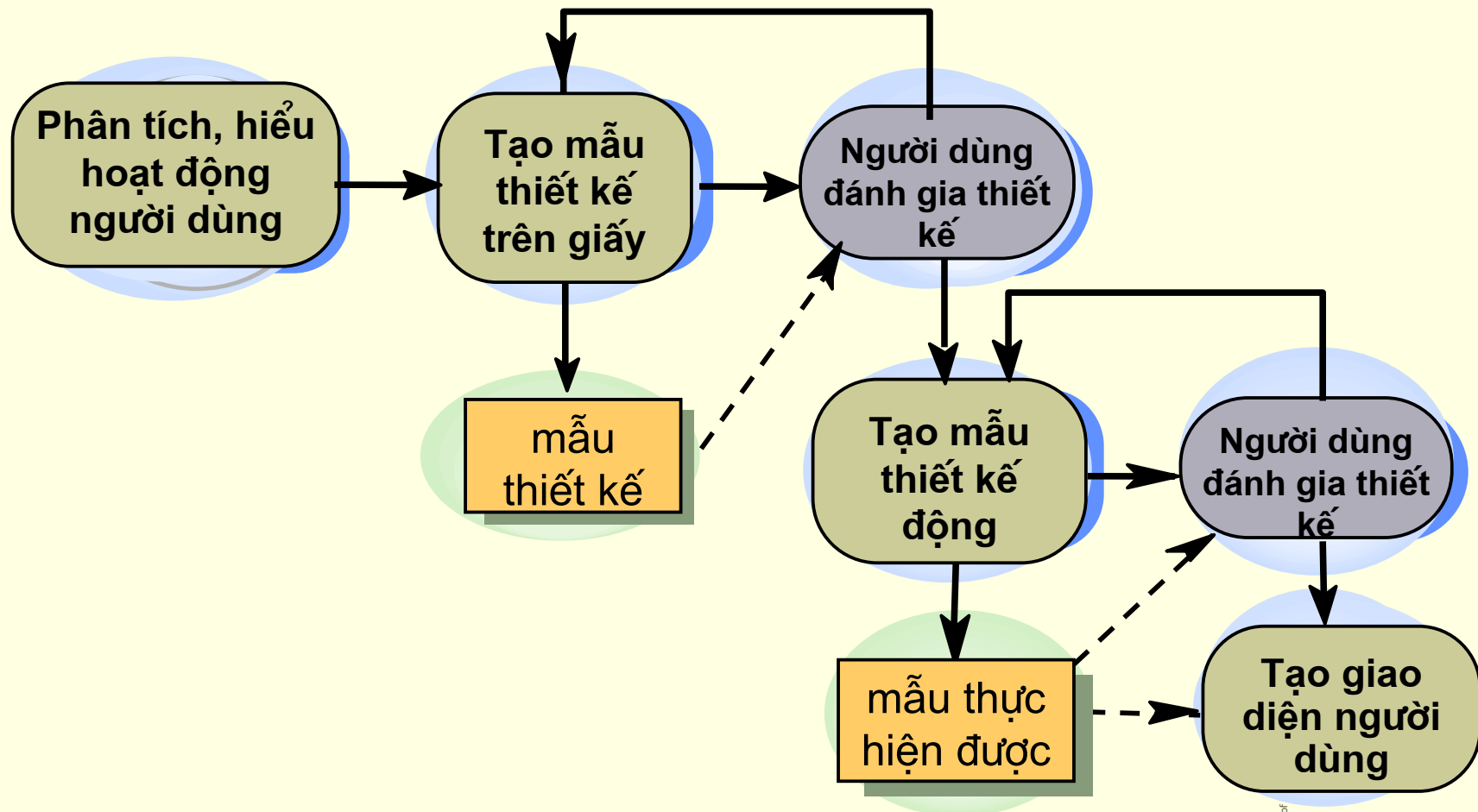


- Là phương tiện để người dùng sử dụng hệ thống
 - ◆ Giao diện thiết kế nghèo nàn người dùng dễ mắc lỗi
 - ◆ Giao diện thiết kế tồi là lý do nhiều phần mềm không được sử dụng
- Giao diện trợ giúp người dùng làm việc với khả năng của họ
 - ◆ Giao diện trợ giúp tốt người dùng thành công
 - ◆ Giao diện trợ giúp tồi, người dùng khó khăn, thất bại

Tiến trình thiết kế giao diện chung



Tiến trình thiết kế giao diện làm mẫu



Nguyên tắc thiết kế giao diện



- Cần phản ánh vào thiết kế:
 - Kinh nghiệm, năng lực, nhu cầu của người dùng
 - ☐ khả năng dùng bàn phím, mouse,
 - ☐ tốc độ phản ứng, khả năng nhớ thao tác
 - Sở thích, văn hóa, lứa tuổi: màu sắc, ngôn ngữ, ..
 - Những hạn chế về mặt vật chất và tinh thần của người dùng (trí nhớ, vụng về, ..có thể mắc lỗi)
- Luôn bao gồm việc làm bản mẫu để người dùng đánh giá

Các nguyên tắc thiết kế giao diện



- Giao diện cần có các tính chất sau đây:
 - **Tính thân thiện:** thuật ngữ, khái niệm, thói quen, trình tự nghiệp vụ của người dùng
 - **Tính nhất quán:** ví tri hiển thị, câu lệnh, thực đơn, biểu tượng, màu sắc, □ cùng dạng
 - **It gây ngạc nhiên**
 - **Có cơ chế phục hồi** tình trạng trước lỗi
 - Cung cấp kịp thời **phản hồi và trợ giúp** mọi lúc, mọi nơi
 - Tiện ích **tương tác đa dạng**

Thiết bị tương tác



■ Thiết bị tương tác thường gặp

- Màn hình
- Bàn phím
- Mouse, bút từ, ...
- Màn hình cảm biến
- Mic/Speaker
- Smart cards, □
- Bóng xoay

Các kiểu tương tác



■ Các kiểu tương tác thông dụng

- Thao tác trực tiếp
- Chọn thực đơn
- Chọn biểu tượng
- Điền vào mẫu biểu
- Ngôn ngữ lệnh
- Ngôn ngữ tự nhiên

Các loại giao diện truyền thống



- ☐ Giao diện dòng lệnh (giao diện hỏi đáp)
- ☐ Giao diện đồ họa (Graphic User Interface - GUI)

Giao diện dòng lệnh



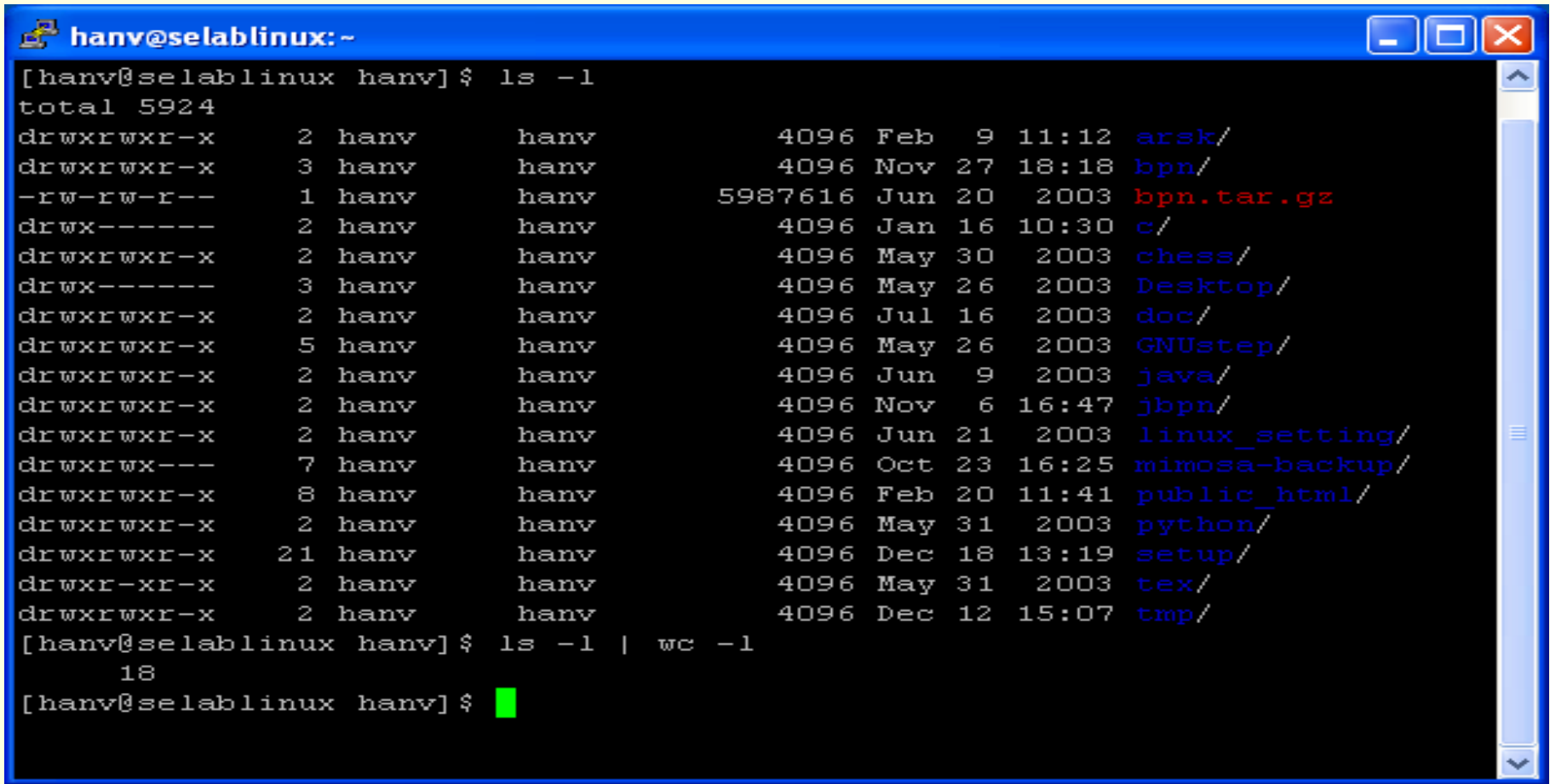
- Là phương thức tương tác có sớm nhất
- Nhập lệnh/dữ liệu từ bàn phím
- Dễ cài đặt so với GUI
 - ◆ thực hiện thông qua hàm chuẩn của ngôn ngữ
 - ◆ không tốn tài nguyên hệ thống
- Có khả năng tổ hợp lệnh để tạo các lệnh phức tạp
 - ◆ phối hợp các filter, tạo các lô xử lý (batch)
 - ◆ có thể lập trình bằng (Unix) shell
 - ◆ có thể tự động hóa

Giao diện dòng lệnh(t)



- Thao tác thực hiện tuần tự
 - ◆ khó sửa lỗi thao tác trước đó
- Không phù hợp với người dùng ít kinh nghiệm
 - ◆ khó học, khó nhớ
 - ◆ dễ nhầm
 - ◆ đòi hỏi kỹ năng sử dụng bàn phím

Giao diện dòng lệnh (t)



A terminal window titled 'hanv@selablinux: ~' with standard window controls. It displays the output of the 'ls -l' command, showing a list of directories and files with their permissions, sizes, dates, and names. The names are color-coded: blue for directories and red for files. The output is as follows:

```
[hanv@selablinux hanv]$ ls -l
total 5924
drwxrwxr-x    2 hanv   hanv      4096 Feb  9 11:12 arsk/
drwxrwxr-x    3 hanv   hanv      4096 Nov 27 18:18 bpn/
-rw-rw-r--    1 hanv   hanv    5987616 Jun 20  2003 bpn.tar.gz
drwx-----    2 hanv   hanv      4096 Jan 16 10:30 c/
drwxrwxr-x    2 hanv   hanv      4096 May 30  2003 chess/
drwx-----    3 hanv   hanv      4096 May 26  2003 Desktop/
drwxrwxr-x    2 hanv   hanv      4096 Jul 16  2003 doc/
drwxrwxr-x    5 hanv   hanv      4096 May 26  2003 GNUstep/
drwxrwxr-x    2 hanv   hanv      4096 Jun  9  2003 java/
drwxrwxr-x    2 hanv   hanv      4096 Nov  6 16:47 jbpn/
drwxrwxr-x    2 hanv   hanv      4096 Jun 21  2003 linux_setting/
drwxrwx---    7 hanv   hanv      4096 Oct 23 16:25 mimosa-backup/
drwxrwxr-x    8 hanv   hanv      4096 Feb 20 11:41 public_html/
drwxrwxr-x    2 hanv   hanv      4096 May 31  2003 python/
drwxrwxr-x   21 hanv   hanv      4096 Dec 18 13:19 setup/
drwxr-xr-x    2 hanv   hanv      4096 May 31  2003 tex/
drwxrwxr-x    2 hanv   hanv      4096 Dec 12 15:07 tmp/
[hanv@selablinux hanv]$ ls -l | wc -l
18
[hanv@selablinux hanv]$
```

Giao diện đồ họa (GUI)



- Thông dụng trên PC, Apple, Unix WS
- Dễ học, dễ sử dụng, thuận tiện với người ít kinh nghiệm
- Có nhiều cửa sổ, có thể tương tác song song
- Hiển thị, tương tác dữ liệu trên nhiều vị trí trong cửa sổ
- Tương tác trực tiếp với thông tin: soạn thảo; nhập dữ liệu vào các form□
 - ◆ dễ học, dễ sử dụng
 - ◆ nhận được tức thời kết quả thao tác
 - ◆ cài đặt phức tạp, tốn tài nguyên phần cứng

Các hình thức tương tác



1. Tương tác trực tiếp và gián tiếp

■ Tương tác gián tiếp

ví dụ: chọn lệnh từ menu, giao diện dòng lệnh

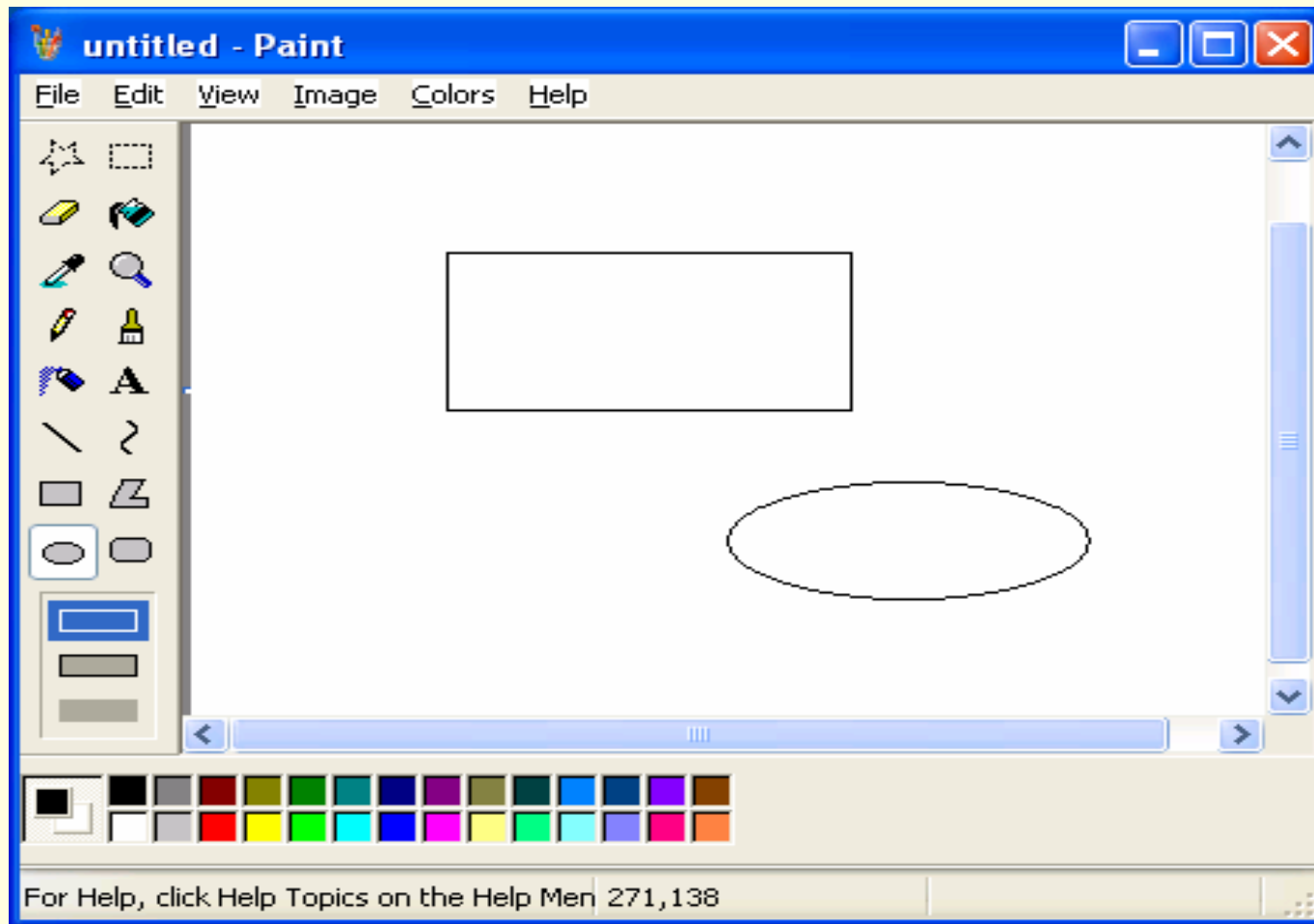
- ◆ kém trực quan

- ◆ thuận tiện khi lặp lại thao tác phức tạp

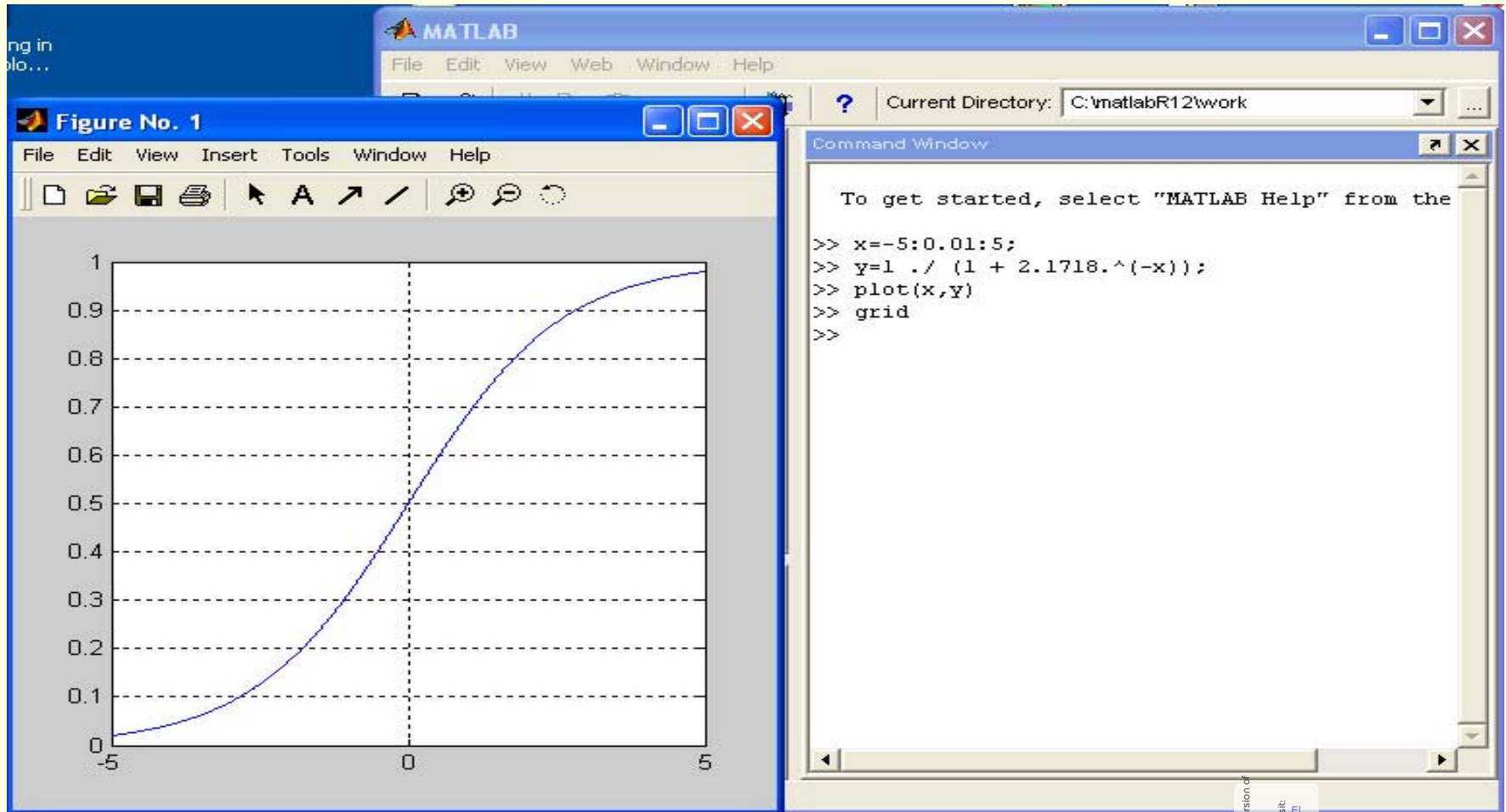
■ Tương tác trực tiếp

- ◆ Dễ nhận biết và thao tác

Ví dụ tương tác trực tiếp



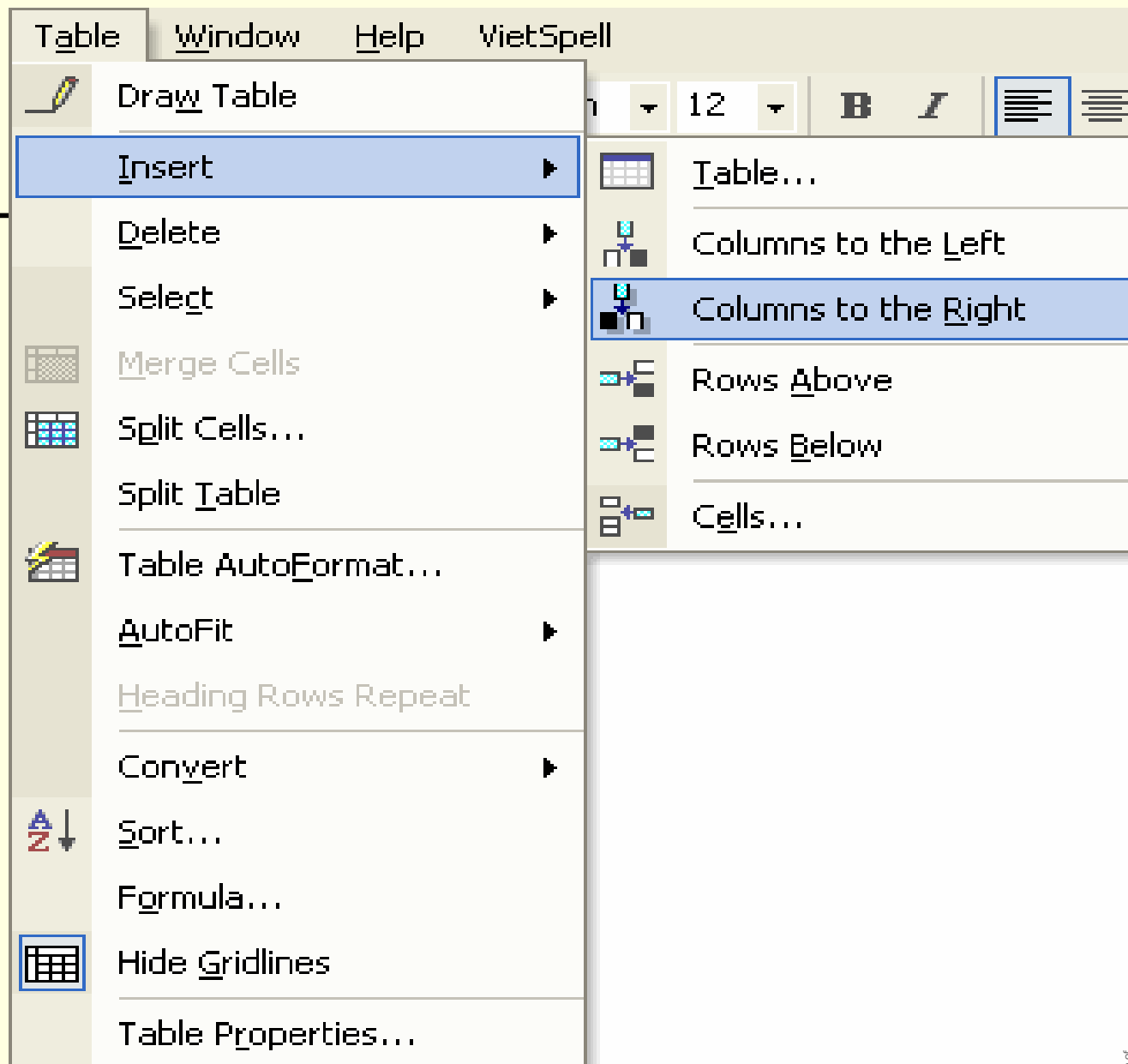
Ví dụ tương tác gián tiếp



Dạng thực đơn



- ◆ Không cần nhớ lệnh
- ◆ Tối thiểu hóa dùng bàn phím
- ◆ Tránh các lỗi như sai lệnh, sai tham số
- ◆ Dễ dàng tạo các trợ giúp theo ngữ cảnh



Một số các vấn đề thiết kế giao diện

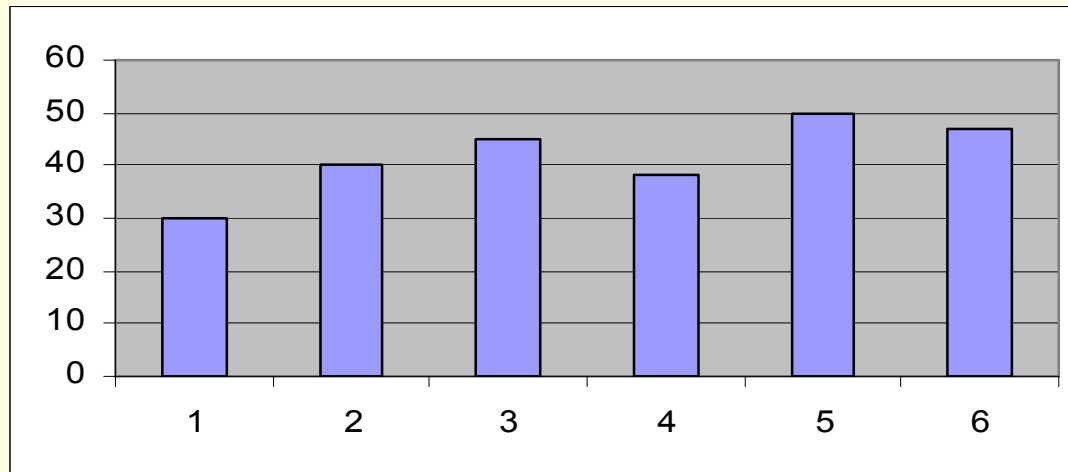


- ◆ Phương pháp hiển thị thông tin
- ◆ Thời gian phản hồi của hệ thống
- ◆ Cách thức xây dựng thông báo
- ◆ Các tiện ích trợ giúp

Hiển thị thông tin



- Hiển thị văn bản (text):
 - chính xác
 - dễ cài đặt
- Hiển thị đồ họa (graphic):- trực quan
 - dễ nhận dạng & mối quan hệ



Thời gian phản hồi



- Thời gian trung bình
 - ◆ thời gian trung bình phản hồi với thao tác
 - ◆ người dùng không thể đợi quá lâu ($< 3s$)
 - ◆ cần chứng tỏ hệ thống đang hoạt động
- Độ biến thiên thời gian
 - ◆ Chênh lệch không được lớn
 - ◆ Đồng đều là tốt nhất

Thông báo



- Phản hồi của hệ thống đối với thao tác
- Có nghĩa, dễ hiểu, các thông tin là hữu ích
 - ◆ tránh đưa ra các số hiệu
 - ◆ định dạng thông báo phải nhất quán(vị trí, nội dung)
- Thông báo lỗi
 - ◆ chính xác
 - ◆ có tính xây dựng (nguyên nhân, cách khắc phục,□)

Thông báo



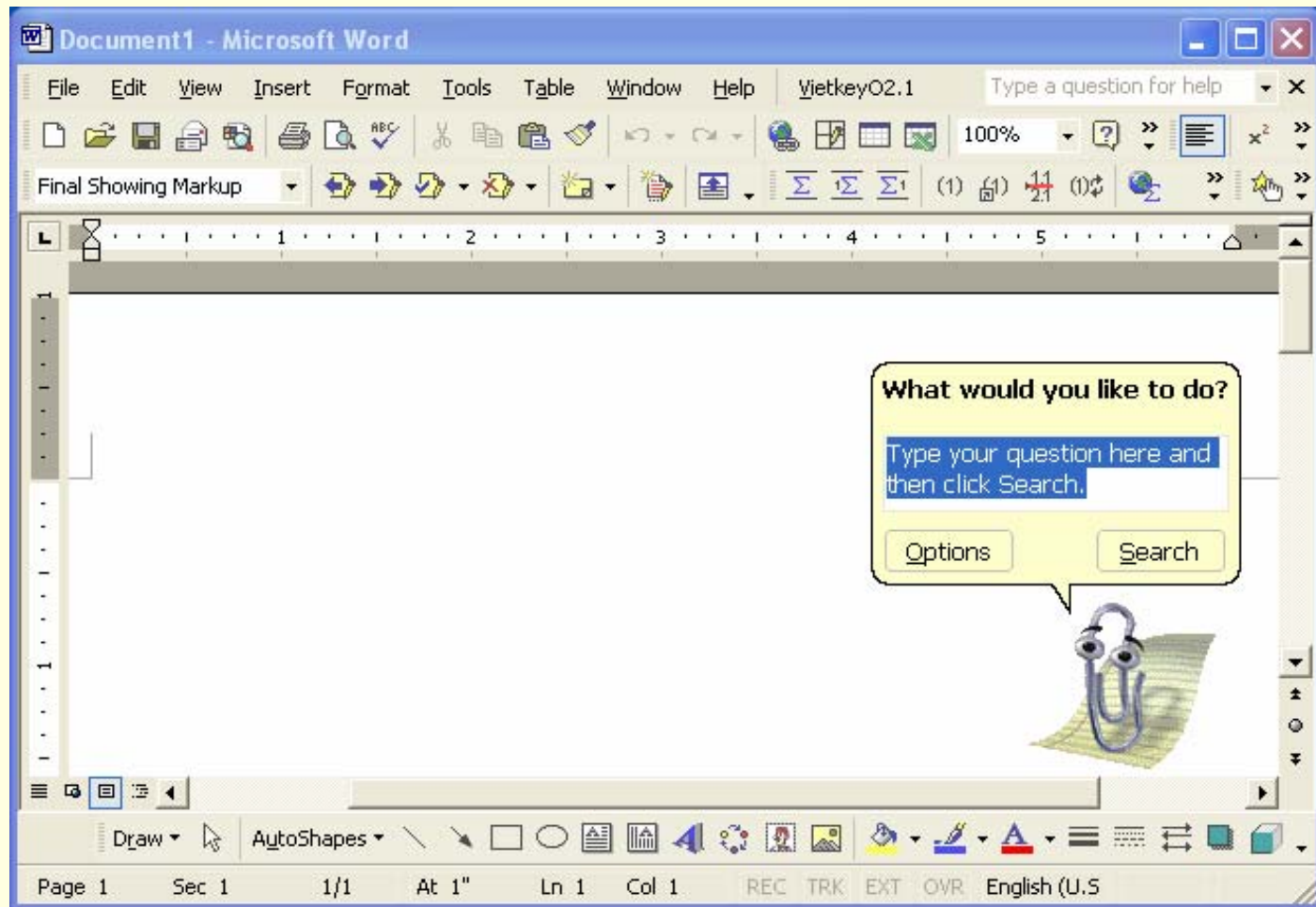
- Số lượng thông báo:
 - ◆ đưa ra càng nhiều càng tốt = càng thân thiện
 - ◆ đưa ra một lượng tối thiểu = im lặng là vàng
- Thời điểm & thứ tự đưa ra thông báo
(phù hợp với cách người dùng)
- Yêu cầu phản hồi đối với thông báo

Tiện ích



- Cần có nhiều các tiện ích trợ giúp khác nhau
- Tiện ích tích hợp: **trợ giúp trực tuyến, theo ngữ cảnh** (chú giải thao tác, giao diện)
- Các tài liệu trực tuyến: **tra cứu chức năng hệ thống**
- Các macro: tự động hóa thao tác: **MS Word macro**

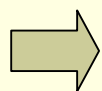
Trợ giúp trực tuyến của MS Office



Tính kỹ nghệ



- Giao diện là phần tử dễ thay đổi
 - ◆ thay đổi qui trình, phương thức thao tác
 - ◆ thay đổi môi trường (phần cứng, hệ điều hành)
 - ◆ nâng cấp (đẹp hơn, dễ sử dụng hơn□)
- Giao diện phải dễ sửa đổi
- Giao diện phải có tính khả chuyển



Giao diện nên độc lập với xử lý thông tin

Một số hình thức cài đặt giao diện

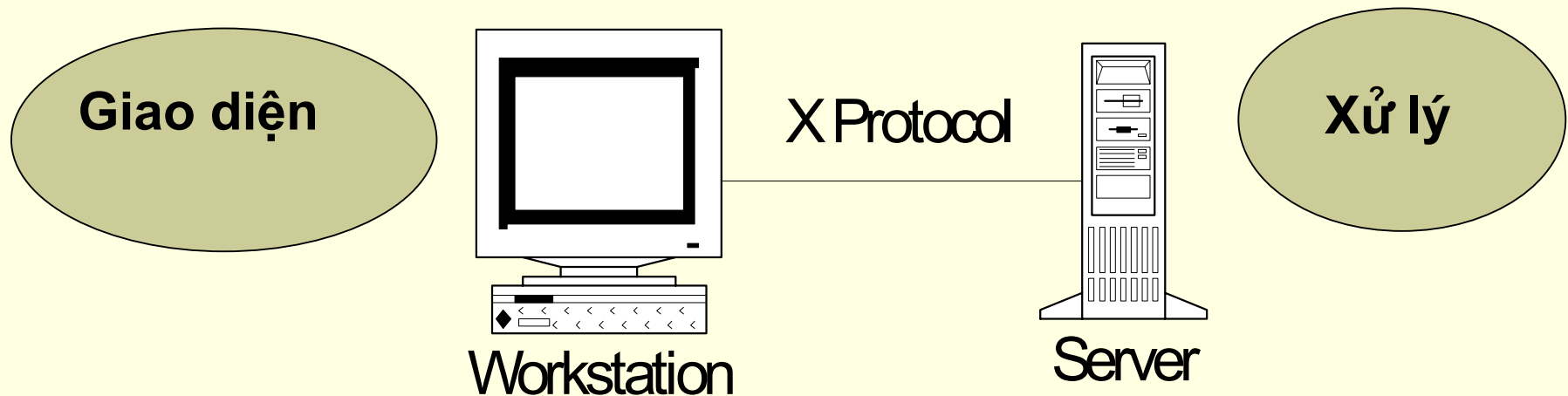


- Tích hợp: dòng lệnh, GUI truyền thống
 - ◆ phát triển bằng cùng ngôn ngữ, cùng bộ công cụ
- Client/Server
 - ◆ giao diện và xử lý là các chương trình độc lập
- X Windows (X protocol)
 - ◆ giao diện và xử lý nằm trong một chương trình
 - ◆ hoạt động phân tán trên mạng

Một số hình thức cài đặt giao diện

■ Web-based

- ◆ truy cập được từ mọi thiết bị có web browser
- ◆ không cần cài đặt thêm phần mềm vào client



Câu hỏi ôn tập



10. Tầm quan trọng của giao diện phần mềm?
11. Những yếu tố người dùng nào cần quan tâm khi thiết kế giao diện?
12. Mô tả tiến trình thiết kế giao diện?
13. Các nguyên tắc thiết kế giao diện?
14. Có những loại thiết bị nào có thể sử dụng để tương tác với hệ thống?
15. Có những loại giao diện nào? Giải thích nội dung, ý nghĩa, ưu nhược điểm của nó?
16. Có những đặc trưng gì cần quan tâm khi thiết kế giao diện? Giải thích nội dung của nó?

Câu hỏi và thảo luận

