

THIẾT KẾ PHẦN MỀM

Lê Thị Mỹ Hạnh
Khoa CNTT
Trường Đại học Bách khoa

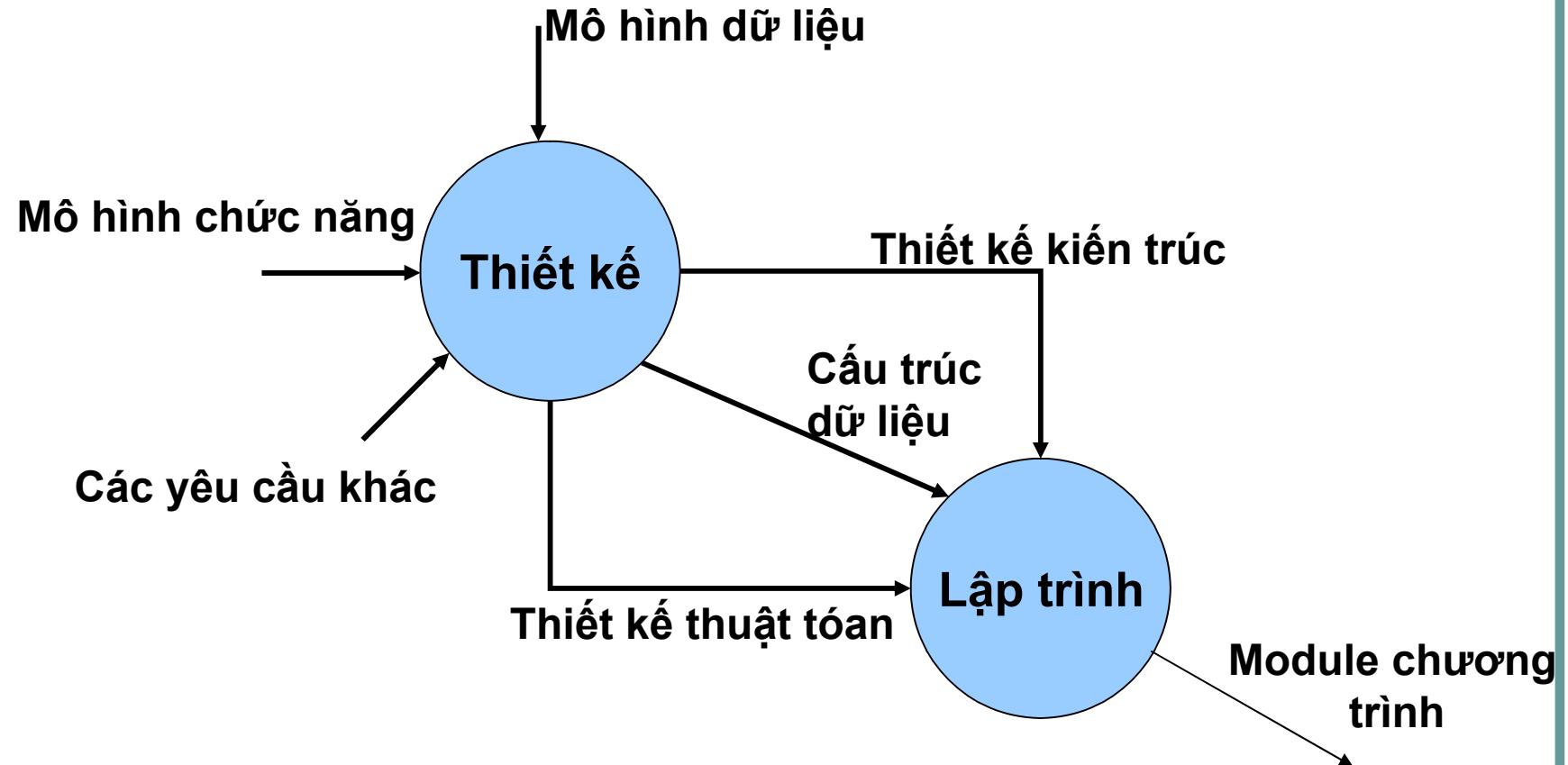
Nội dung

- Khái niệm, nguyên lý thiết kế phần mềm
- Chất lượng thiết kế
- Thiết kế kiến trúc
- Thiết kế giao diện
- Một số phương pháp và công cụ thiết kế

Thiết kế phần mềm – khái niệm

- Thiết kế phần mềm là pha tiếp theo pha phân tích
 - Công việc của người phát triển
 - Tìm giải pháp cho yêu cầu (How to do)
- ➡ *Là quá trình sáng tạo*
 - Nghĩ xem nên làm thế nào
 - Biểu diễn cách thức phương án
 - Xem xét lại, chi tiết hóa
- Đủ chi tiết để lập trình được

Thiết kế phần mềm – khái niệm



Thiết kế phần mềm – vai trò

- Tạo mô hình cài đặt của phần mềm
 - Là phương tiện trao đổi thông tin để đảm bảo chất lượng
 - dễ hiểu, dễ sửa đổi hơn mã chương trình
 - có nhiều mức chi tiết; cung cấp cái nhìn tổng thể
- Nếu không có thiết kế; hoặc thiết kế tồi
 - làm tăng công sức mã hóa
 - làm tăng công sức bảo trì

Thiết kế phần mềm – mục tiêu

- Tính mở
- Tính dễ thay đổi
- Dễ hiểu, dễ triển khai (đủ chi tiết)
- Hiệu quả về tốc độ, kích cỡ

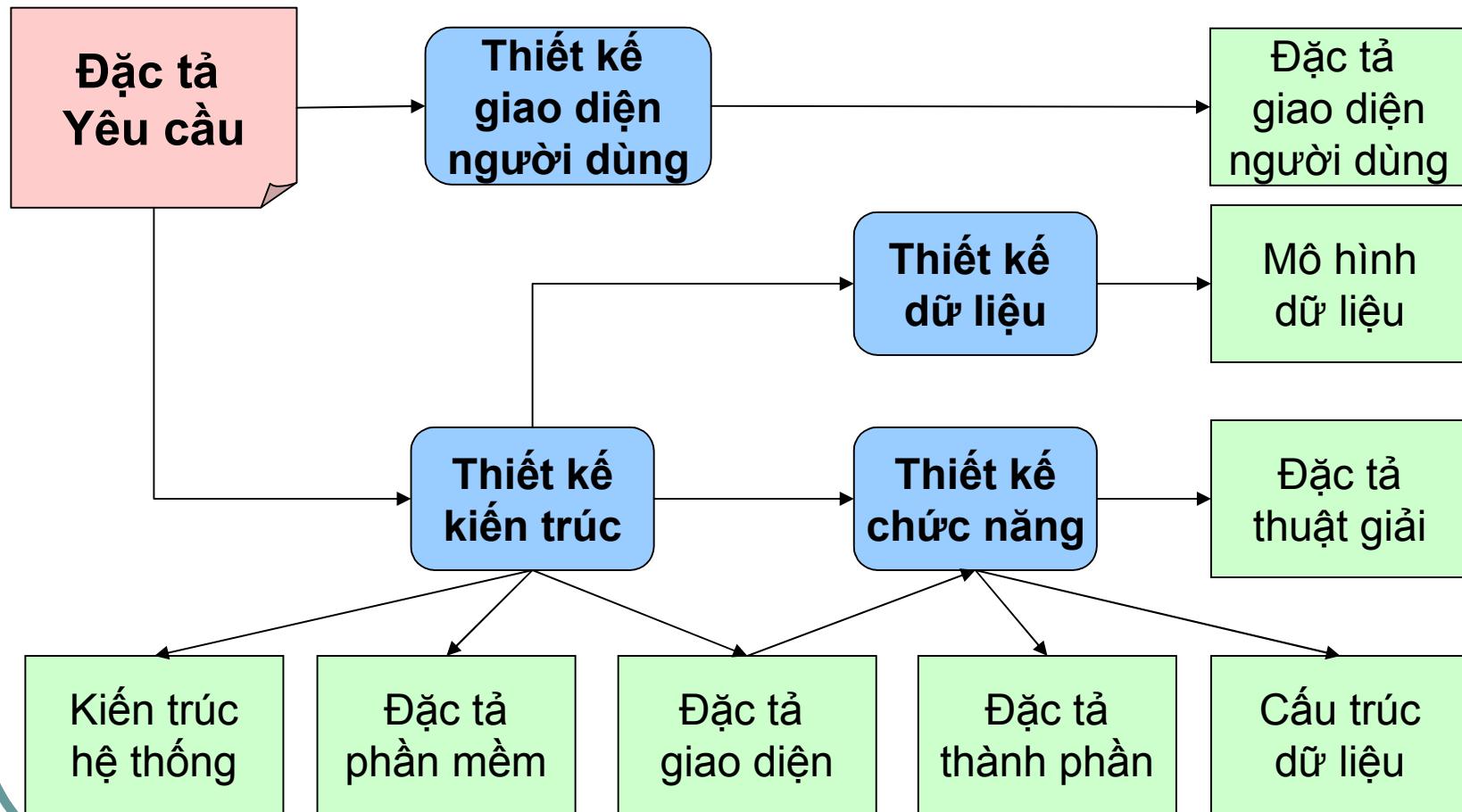
Thiết kế phần mềm – nguyên lý

- Thiết kế không nên bị bó buộc vào cái nhìn hạn hẹp
 - nên được lựa chọn từ các giải pháp khác nhau
- Thiết kế phải lần ngược lại được mô hình phân tích
 - các mô đun và các yêu cầu không phải là tương ứng một một
 - phải kiểm tra được sự thỏa mãn các yêu cầu
- Không nên tạo lại các thiết kế (giải pháp) đã có
 - cần tái sử dụng tối đa các thiết kế đã có
- Mô hình thiết kế (giải pháp) nên tiến gần đến mô hình thế giới thực (bài toán)
- Biểu diễn thiết kế phải có tính nhất quán và tính tích hợp
 - thiết kế do nhiều người tiến hành song song
 - phải thống nhất cách biểu diễn, thống nhất giao diện

Thiết kế phần mềm – nguyên lý (2)

- Thiết kế cần có cấu trúc để dễ dàng thay đổi
 - phải được môđun hóa, phân cấp
- Thiết kế không phải là mã hóa
 - thiết kế luôn có mức trừu tượng hơn mã hóa, đảm bảo dễ hiểu, dễ thay đổi
- Thiết kế cần được đánh giá chất lượng ngay trong khi nó được tạo ra
 - tính kết dính, tính ghép nối, hiệu quả thuật toán...
- Thiết kế cần được thẩm định để tránh các lỗi mang tính hệ thống
 - thiếu chức năng, chức năng không rõ ràng, mâu thuẫn...

Thiết kế phần mềm - các hoạt động



Thiết kế phần mềm - các hoạt động

- Thiết kế kiến trúc

- phân rã hệ thống thành các mô đun, hệ thống con
- xác định tương tác (giao diện) giữa các mô đun
- biểu diễn bằng biểu đồ cấu trúc (structure chart)

⇒ *chưa cân chỉ ra chi tiết*

- Thiết kế dữ liệu

- xây dựng mô hình biểu diễn thông tin
- thiết kế dữ liệu lô gich
- thiết kế dữ liệu vật lý (kiểu dữ liệu, phi chuẩn)

⇒ *Ảnh hưởng mạnh đến chất lượng phần mềm*

Thiết kế phần mềm - các hoạt động

- Thiết kế chức năng
 - xác định các mô đun, tương tác giữa chúng
 - cấu trúc dữ liệu, thuật toán xử lý
 - cách biểu diễn: biểu đồ luồng, giả mă,...
- Thiết kế giao diện người dùng
 - nên nhìn nhận giao diện là một bài toán độc lập
 - có thể tiến hành sớm ở pha phân tích

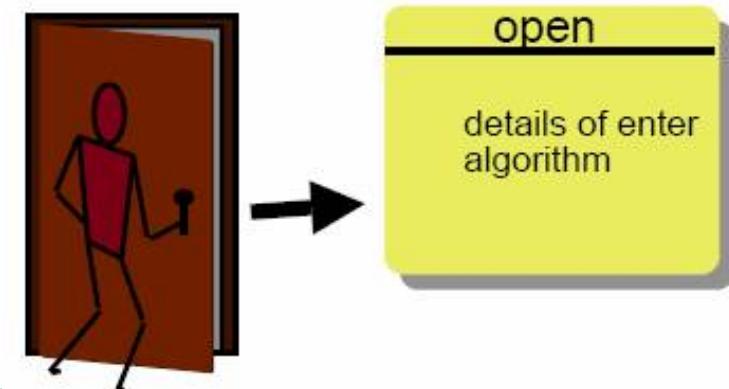
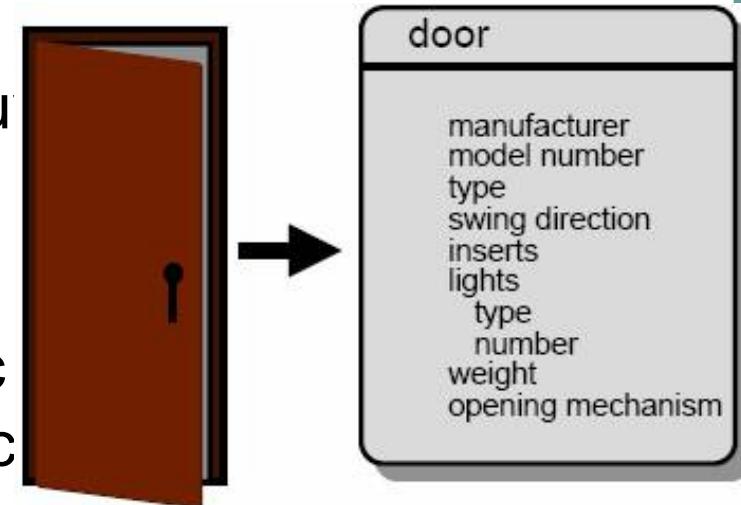
Thiết kế phần mềm – các khái niệm

- **Trùu tượng hóa** - trùu tượng hóa dữ liệu, thủ tục, điều khiển
- **Làm mịn** - chi tiết hóa và bổ sung (tính sáng tạo)
- **Tính mô đun** - phân chia dữ liệu và chức năng
- **Kiến trúc** - cấu trúc tổng thể của phần mềm
- **Thủ tục** - thuật toán để thực hiện chức năng
- **Che dấu** - điều khiển bằng giao diện

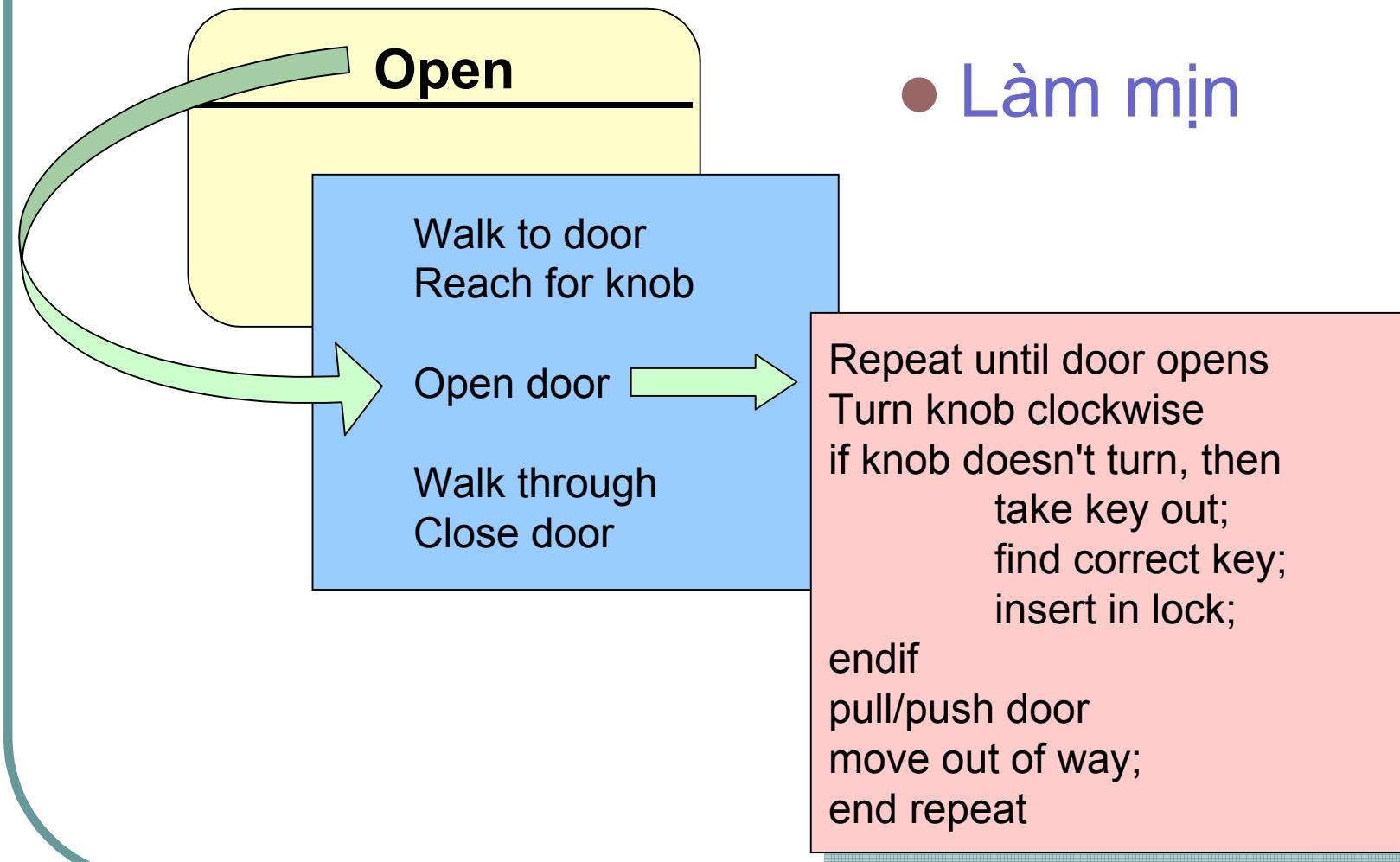
Các khái niệm

Trùu tượng hóa

- Là khái niệm cơ sở trong tư duy của con người
- Là quá trình ánh xạ một sự vật/hiện tượng của thế giới thực thành một khái niệm logic
- Có nhiều mức trùu tượng khác nhau
 - cho phép con người tập trung (tư duy) vào giải quyết vấn đề mà không cần bận tâm đến chi tiết
 - biểu diễn vấn đề bằng một cấu trúc tự nhiên



Các khái niệm



Các khái niệm

Thiết kế môđun

- Ý nghĩa
 - Dễ xây dựng, dễ thay thế, sửa đổi

- Nguyên lý của mô đun hóa
 - Quan điểm chia để trị

$$C(p_1 + p_2) > C(p_1) + C(p_2)$$

$$E(p_1 + p_2) > E(p_1) + E(p_2)$$

Trong đó: C: độ phức tạp

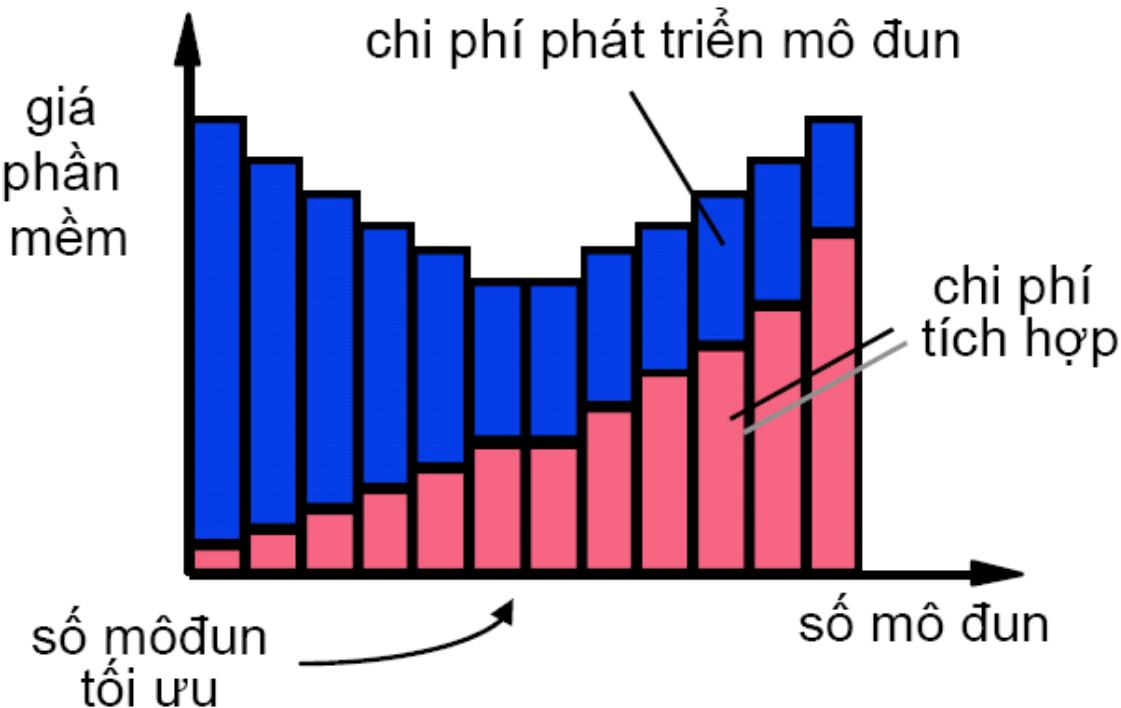
E: nỗ lực thực hiện

- giảm độ phức tạp, cục bộ, dễ sửa đổi
- có khả năng phát triển song song
- dễ sửa đổi, dễ hiểu nên dễ tái sử dụng

Các khái niệm

Thiết kế môđun

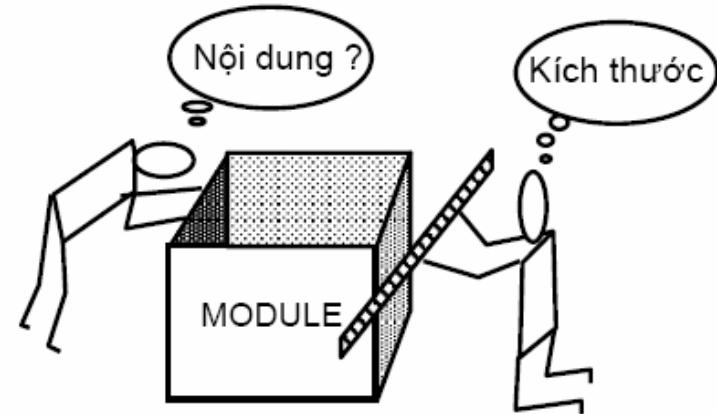
- Số lượng môđun => cần xác định số môđun tối



Các khái niệm

Thiết kế môđun

- Kích cỡ môđun xác định dựa trên khái niệm độc lập chức năng
 - Dễ hiểu, dễ sửa đổi, dễ tái sử dụng



Các khái niệm

Che dấu thông tin

- Sử dụng môđun thông qua các giao diện
 - danh sách tham số và giá trị trả lại
- Không cần biết cách thức cài đặt thực tế
 - thuật toán
 - cấu trúc dữ liệu
 - giao diện ngoại lai (các mô đun thứ cấp, thiết bị vào/ra)
 - tài nguyên hệ thống

Các khái niệm

- **Ý nghĩa che dấu thông tin**
 - Giảm hiệu ứng phụ khi sửa đổi módun
 - Giảm sự tác động của thiết kế tổng thể lên thiết kế cục bộ
 - Nhấn mạnh việc trao đổi thông tin thông qua giao diện
 - Loại bỏ việc sử dụng dữ liệu dùng chung
 - Hướng tới sự đóng gói chức năng – thuộc tính của thiết kế tốt

➡ *Tạo ra các sản phẩm phần mềm tốt hơn*

Chất lượng thiết kế

- Tiêu chí
 - Tính mô đun hóa
 - Tính che dấu thông tin
 - Độ đo chất lượng thiết kế
 - Mức ghép nối giữa các mô đun (*coupling*)
 - Độ kết dính thành phần trong mô đun (*cohesion*)
 - Tính hiểu được (understandability)
 - Tính thích nghi được (adaptability)
- ➡ *Phụ thuộc bài toán, không có phương pháp tổng quát*

Chất lượng thiết kế

- *Ghép nối – độ đo sự liên kết giữa các module*

- mức độ quan hệ của các module
- module nên ghép nối lỏng lẻo
- càng lỏng lẻo càng dễ sửa đổi thiết kế

normal coupling

data coupling

stamp coupling

control coupling

common coupling

content coupling

loose and best

still very good

ok

ok

very bad

tight and worst

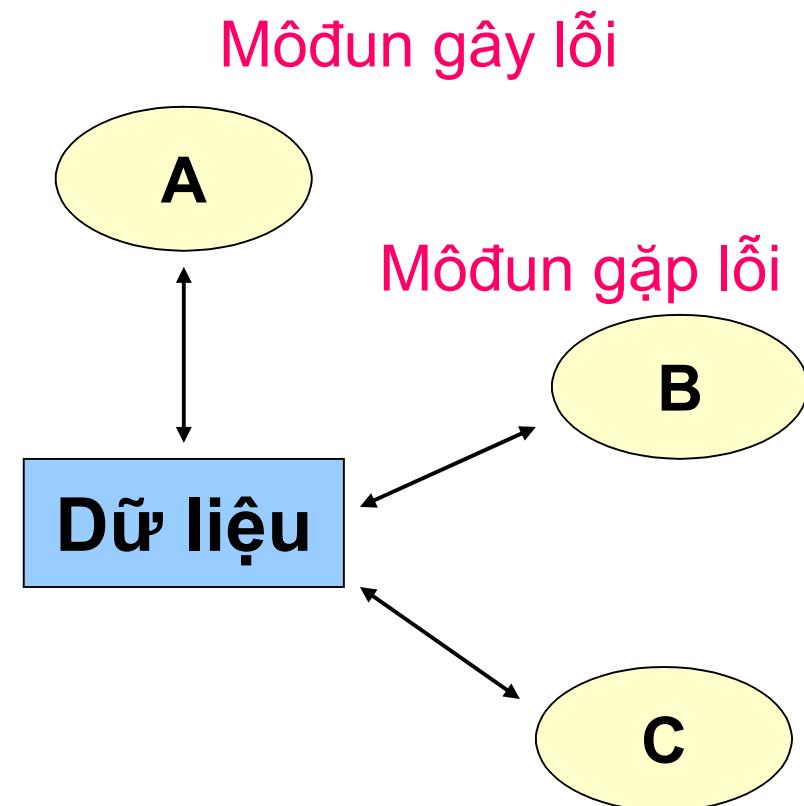
Chất lượng thiết kế - Ghép nối

- **Ghép nối nội dung (content coupling)**
 - là trường hợp xấu nhất
 - các module dùng lẫn dữ liệu của nhau
 - các ngôn ngữ bậc thấp không có biến cục bộ
 - lạm dụng lệnh Goto
- Ví dụ 1:
 - Module p thay đổi một câu lệnh của module q
- Ví dụ 2:
 - Module p tham chiếu đến dữ liệu cục bộ của module q dưới dạng sự thay thế số bên trong q
- Ví dụ 3:
 - Module p phân thành một nhãn cục bộ trong module q

Chất lượng thiết kế - Ghép nối

- Ghép nối chung
(common coupling)

- Các module trao đổi dữ liệu thông qua biến tổng thể
- Lỗi của module này có thể ảnh hưởng đến hoạt động của module khác
- Khó sử dụng lại các module
- Ví dụ: Modules B và C truy cập cùng một CSDL và cả hai có thể *read* và *write* cùng một bản ghi



Chất lượng thiết kế - Ghép nối

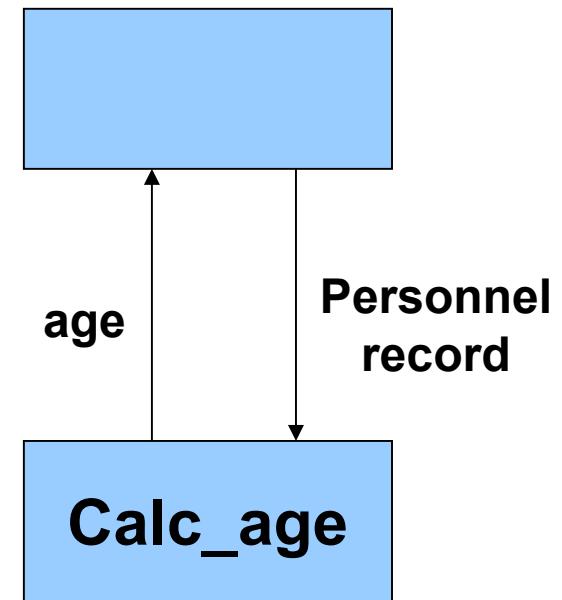
- Ghép nối điều khiển (control coupling)
 - Các module trao đổi thông tin điều khiển
 - Làm cho thiết kế khó hiểu, khó sửa đổi, dễ nhầm

```
procedure PrintRec is
begin
  DisplayName (name, sex);
  ....
end PrintRec;
```

```
procedure DisplayName (in : name, sex) is
begin
  if sex = m
  then
    print Mr.
  else
    print Ms
  print name
end DisplayName;
```

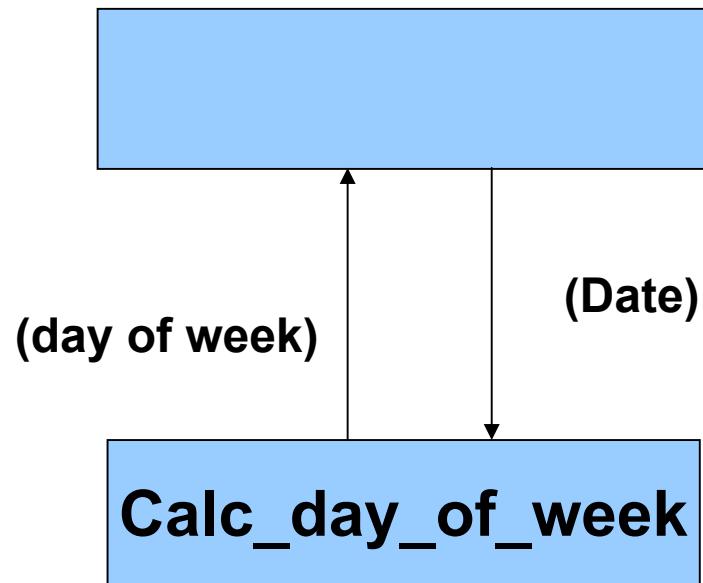
Chất lượng thiết kế - Ghép nối

- Ghép nối nhãn (stamp coupling)
 - Các module trao đổi thông tin
 - Module có thể thực hiện chức năng ngoài ý muốn
 - Làm giảm tính thích nghi



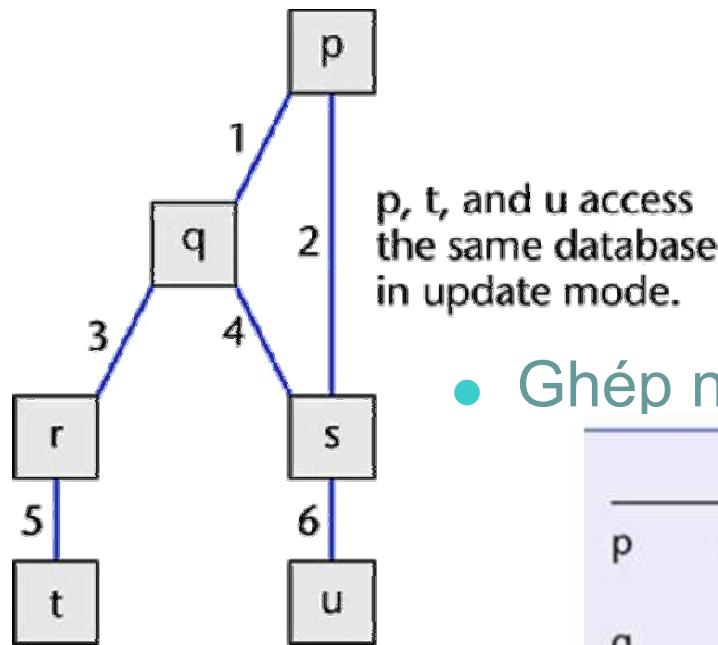
Chất lượng thiết kế - Ghép nối

- Ghép nối dữ liệu (data coupling)
 - Truyền dữ liệu qua tham số
 - Nhận kết quả qua tham số và giá trị trả lại



Chất lượng thiết kế - Ghép nối

- Ví dụ



- Mô tả giao diện

Number	In	Out
1	aircraft_type	status_flag
2	list_of_aircraft_parts	—
3	function_code	—
4	list_of_aircraft_parts	—
5	part_number	part_manufacturer
6	part_number	part_name

- Ghép nối giữa tất cả các cặp module

	q	r	s	t	u
p	Data	—	{ Data or stamp	Common	Common
q		Control	{ Data or stamp	—	—
r			—	Data	—
s				—	Data
t					Common

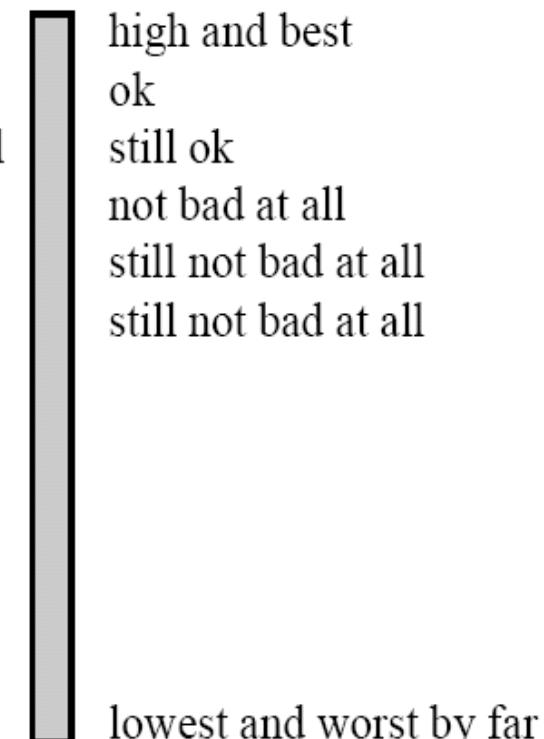
Chất lượng thiết kế - Kết dính (Cohesion)

- **Kết dính** – độ đo sự phụ thuộc lẫn nhau của các thành phần trong một môđun

- mỗi module chỉ nên thực hiện một chức năng
- mọi thành phần nên tham gia thực hiện chức năng đó
- kết dính cao thì tính cục bộ cao (độc lập chức năng);
- dễ hiểu, dễ sửa đổi

functional
sequential
communicational
procedural
temporal
logical

coincidental



Chất lượng thiết kế - Kết dính

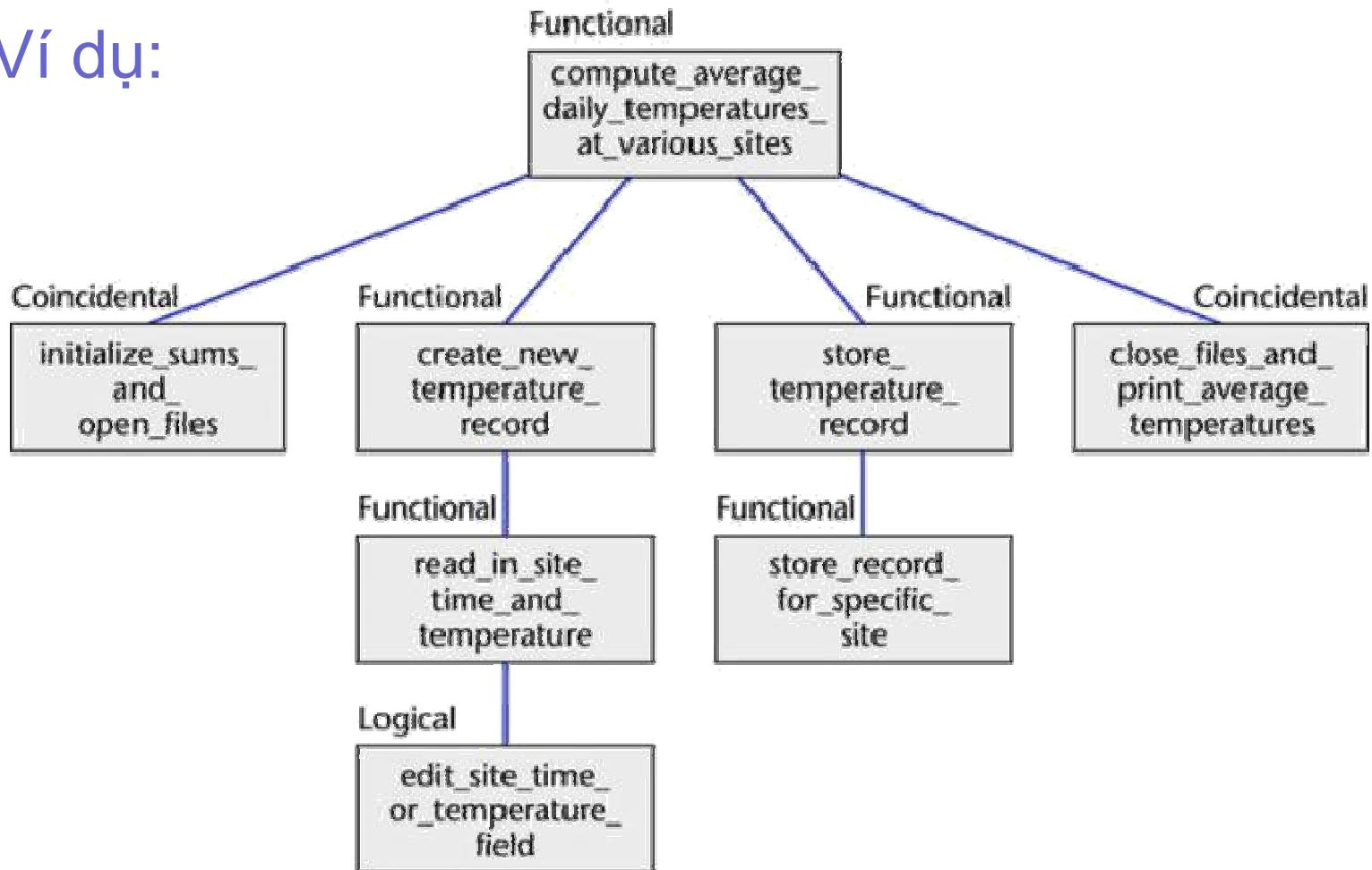
- **Kết dính gom góp (coincidental cohesion)**
 - các thành phần không liên quan đến nhau
 - Vd: print_next_line, reverse_string_of_characters_comprising_second_parameter, add_7_to_fifth_parameter, convert_fourth_parameter_to_floating_point
- **Kết dính lôgic (logical cohesion)**
 - các thành phần làm chức năng lôgic tương tự
 - vd: hàm xử lý lỗi chung
- **Kết dính thời điểm (temporal cohesion)**
 - các thành phần hoạt động cùng thời điểm
 - vd: hàm khởi tạo (đọc dữ liệu, cấp phát bộ nhớ)

Chất lượng thiết kế - Kết dính

- **Kết dính thủ tục (procedural cohesion)**
 - các thành phần tạo có một thứ tự xác định
 - vd: tính lương cơ bản, tính phụ cấp, tính bảo hiểm
- **Kết dính truyền thông (communicational cohesion)**
 - các thành phần truy cập cùng dữ liệu
 - vd: thống kê (tính max, min, mean, variation...)
- **Kết dính tuần tự (sequential cohesion)**
 - output của một thành phần là input của thành phần tiếp theo
 - vd: ảnh màu -> đen trắng -> ảnh nén
- **Kết dính chức năng (functional cohesion)**
 - các thành phần cùng góp phần thực hiện một chức năng
 - vd: sắp xếp

Chất lượng thiết kế - Kết dính

Ví dụ:



Chất lượng thiết kế - Tính hiểu được

- **Tính hiểu được (Understandability):**

- Ghép nối lỏng lẻo
- Kết dính cao
- Được lập tài liệu
- Thuật toán, cấu trúc dễ hiểu

Chất lượng thiết kế - Tính thích nghi

- **Tính thích nghi (Adaptability):**

- Hiểu được

- sửa đổi được, tái sử dụng

- Tự chữa

- không sử dụng thư viện ngoài

- mâu thuẫn với xu hướng tái sử dụng

Chất lượng thiết kế

- Thiết kế hướng đối tượng hướng tới chất lượng thiết kế tốt
 - đóng gói, che dấu thông tin
 - là các thực thể hoạt động độc lập
 - trao đổi dữ liệu qua thông báo
 - có khả năng kế thừa
 - cục bộ, dễ hiểu, dễ tái sử dụng

THIẾT KẾ KIẾN TRÚC – Khái niệm

- Thiết kế kiến trúc là quá trình xác định các hệ thống con của hệ thống và cơ cấu tổ chức (framework) việc điều khiển giao tiếp giữa các hệ thống con
- Kết quả ra của tiến trình thiết kế kiến trúc là sự mô tả về kiến trúc phần mềm

THIẾT KẾ KIẾN TRÚC – Khái niệm

Hệ thống con và các mô đun

- **Hệ thống con** là một hệ thống trong hệ thống tổng thể mà các hoạt động của nó độc lập với các dịch vụ được cung cấp từ các hệ thống con khác
- **Mô đun** là một thành phần hệ thống cung cấp các dịch vụ cho các thành phần khác nhưng không được xem như một hệ thống riêng biệt

THIẾT KẾ KIẾN TRÚC – Khái niệm

Đặc điểm thiết kế kiến trúc phần mềm

- Là giai đoạn đầu của quá trình thiết kế
- Biểu diễn sự kết nối giữa đặc tả yêu cầu và các tiến trình thiết kế
- Thường tiến hành song song với các hoạt động đặc tả phần mềm
- Bao gồm việc xác định các thành phần hệ thống và giao tiếp giữa chúng

THIẾT KẾ KIẾN TRÚC – Khái niệm

Lợi ích của thiết kế kiến trúc phần mềm

- Giao tiếp giữa khách hàng và nhà phát triển
 - Kiến trúc phần mềm là tiêu điểm cho sự thảo luận giữa khách hàng và nhà phát triển
- Phân tích hệ thống
 - Là phương tiện để phân tích khả thi về các yêu cầu phi chức năng của hệ thống
- Sử dụng lại
 - Kiến trúc này có thể được sử dụng lại cho nhiều hệ thống

THIẾT KẾ KIẾN TRÚC – Tiến trình

- 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
- Mô hình hóa điều khiển
 - xác lập mô hình điều khiển giữa các phần của hệ thống
- Phân rã module
 - phân rã các hệ con thành các module

THIẾT KẾ KIẾN TRÚC – Mô hình

Mô hình kiến trúc

- Trong quá trình thiết kế có thể tạo ra nhiều mô hình kiến trúc khác nhau
- Mỗi mô hình biểu diễn các khía cạnh khác nhau của kiến trúc này

THIẾT KẾ KIẾN TRÚC – Mô hình

Các mô hình kiến trúc

- Mô hình kiến trúc tĩnh chỉ ra các thành phần chính của hệ thống
- Mô hình tiến trình động chỉ ra cấu trúc động của hệ thống
- Mô hình giao diện (interface) xác định các giao diện của hệ thống con
- Mô hình các mối quan hệ chỉ ra quan hệ giữa các hệ thống con. Vd: mô hình luồng dữ liệu

THIẾT KẾ KIẾN TRÚC – Mô hình

Các kiểu kiến trúc

- Mô hình kiến trúc của một hệ thống có thể chuyển thành một mô hình kiến trúc tổng quát – kiểu kiến trúc (mô hình kiến trúc chung)
- Kiểu kiến trúc có thể làm đơn giản hóa vấn đề xác định các kiến trúc hệ thống
- Tuy nhiên, hầu hết các hệ thống lớn là hỗn tạp, không tuân theo một kiểu kiến trúc đơn lẻ nào

THIẾT KẾ KIẾN TRÚC – Mô hình

Khuyến cáo kiến trúc phần mềm tốt

- **Hiệu suất**
 - Cục bộ hóa xử lý để tối thiệu hóa giao tiếp giữa các hệ thống con
- **An ninh**
 - Sử dụng kiến trúc phân tầng để che dấu
- **An toàn**
 - Cô lập các thành phần ít an toàn
- **Tính sẵn có**
 - Đưa vào những thành phần chưa cần vào kiến trúc
- **Có thể bảo trì được**
 - Các thành phần độc lập, ghép nối thành phần

THIẾT KẾ KIẾN TRÚC

Thiết kế kiến trúc

- là quá trình xác định **các hệ thống con** của hệ thống và cơ cấu tổ chức (framework) việc **điều khiển giao tiếp** giữa các hệ thống con

Mô hình kiến trúc

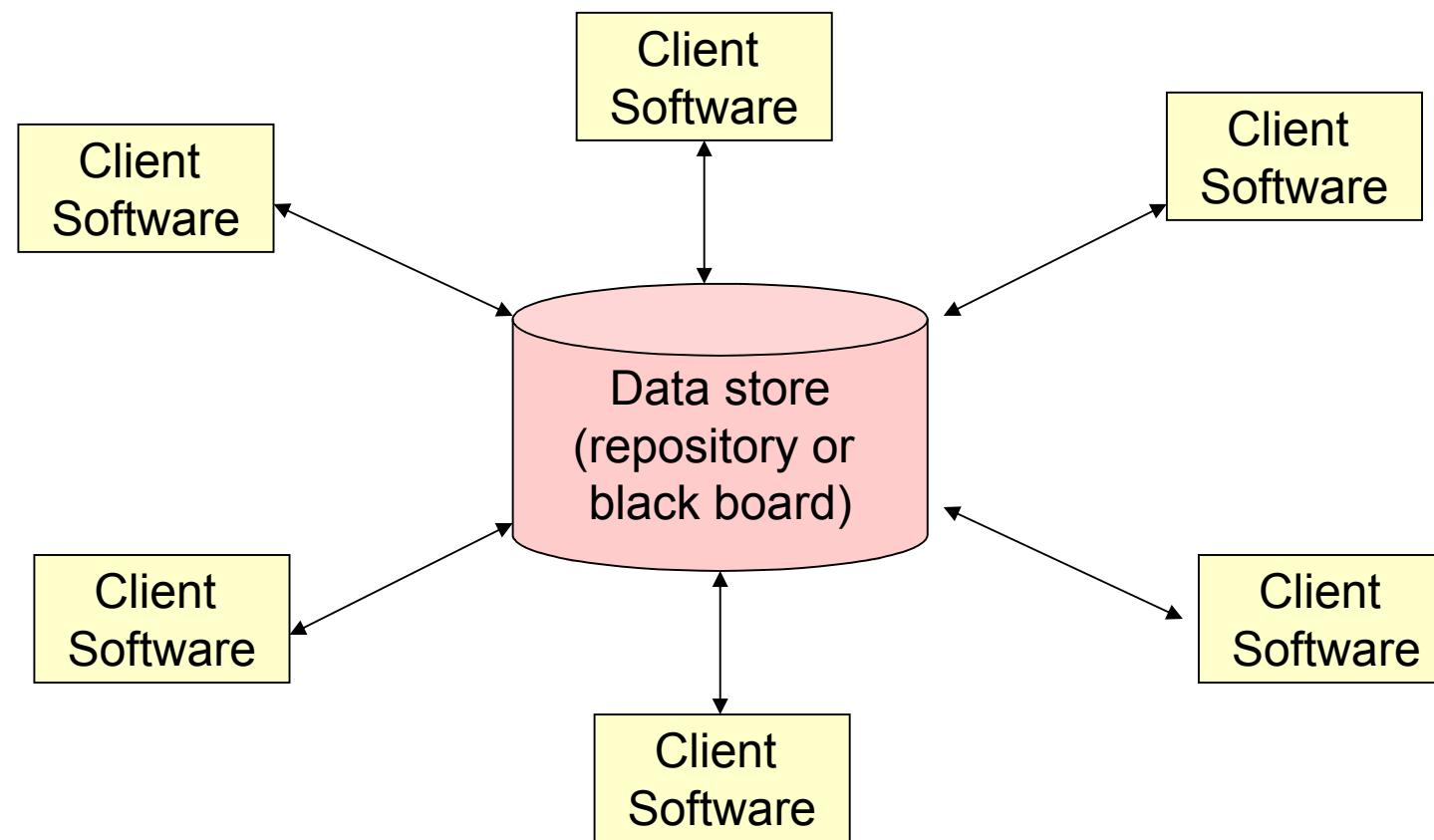
- Trong quá trình thiết kế có thể tạo ra nhiều mô hình kiến trúc khác nhau
- Mỗi mô hình biểu diễn các khía cạnh khác nhau của kiến trúc này

MÔ HÌNH KIẾN TRÚC

- Mô hình biểu diễn một khía cạnh của kiến trúc
 - Mô hình cấu trúc hệ thống
 - Mô hình điều khiển hệ thống
 - Mô hình phân rã mô đun

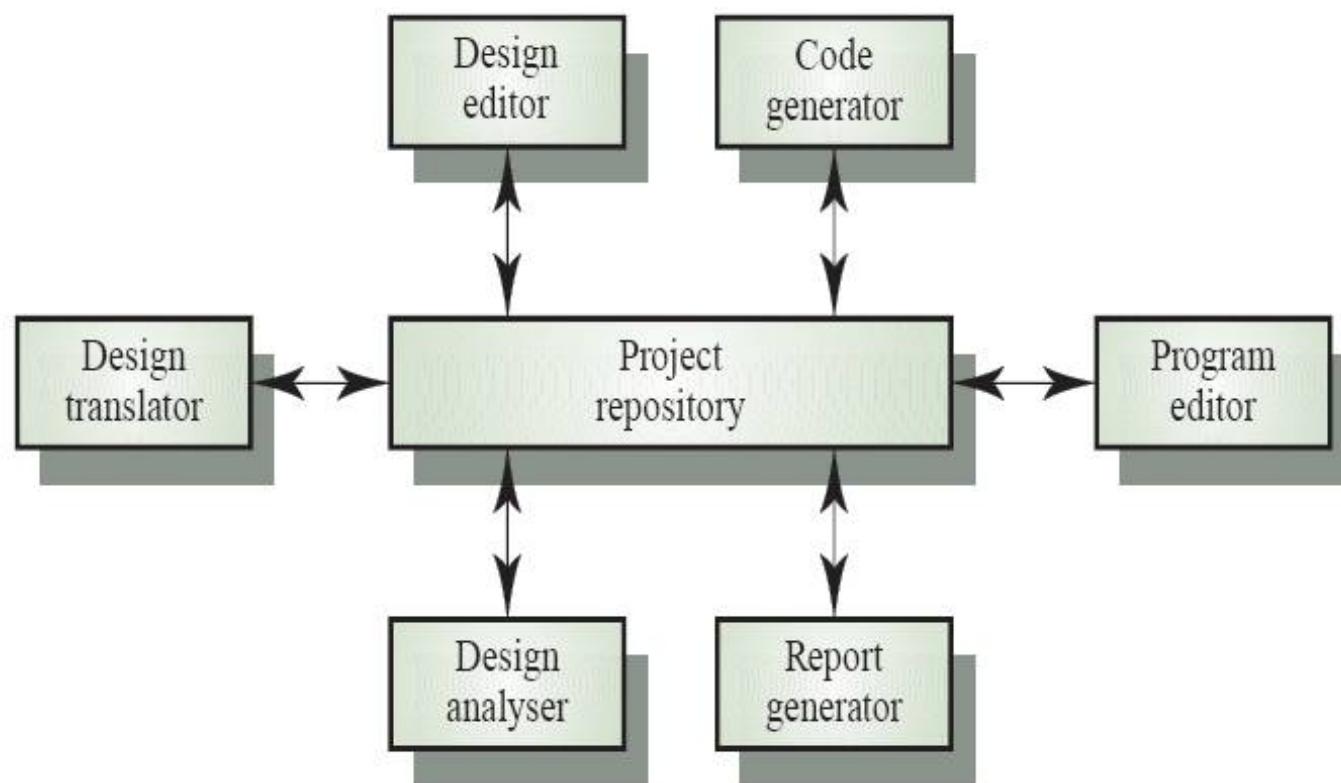
MÔ HÌNH KIẾN TRÚC- Mô hình cấu trúc

Kiến trúc dữ liệu tập trung (Data-Centered Architecture)



MÔ HÌNH KIẾN TRÚC- Mô hình cấu trúc

Ví dụ kiến trúc dữ liệu tập trung: CASE Toolset



MÔ HÌNH KIẾN TRÚC- Mô hình cấu trúc

Kiến trúc dữ liệu tập trung

• Ưu điểm

- Tiện lợi cho chia sẻ dữ liệu lớn
- Các phân hệ không cần quan tâm đến tổ chức dữ liệu

• Nhược điểm

- Các phân hệ phải thống nhất mô hình dữ liệu chung
- Khó thay đổi cấu trúc dữ liệu
- Các phân hệ không thể đưa ra chính sách riêng
- Khó khăn trong quản lý giao dịch

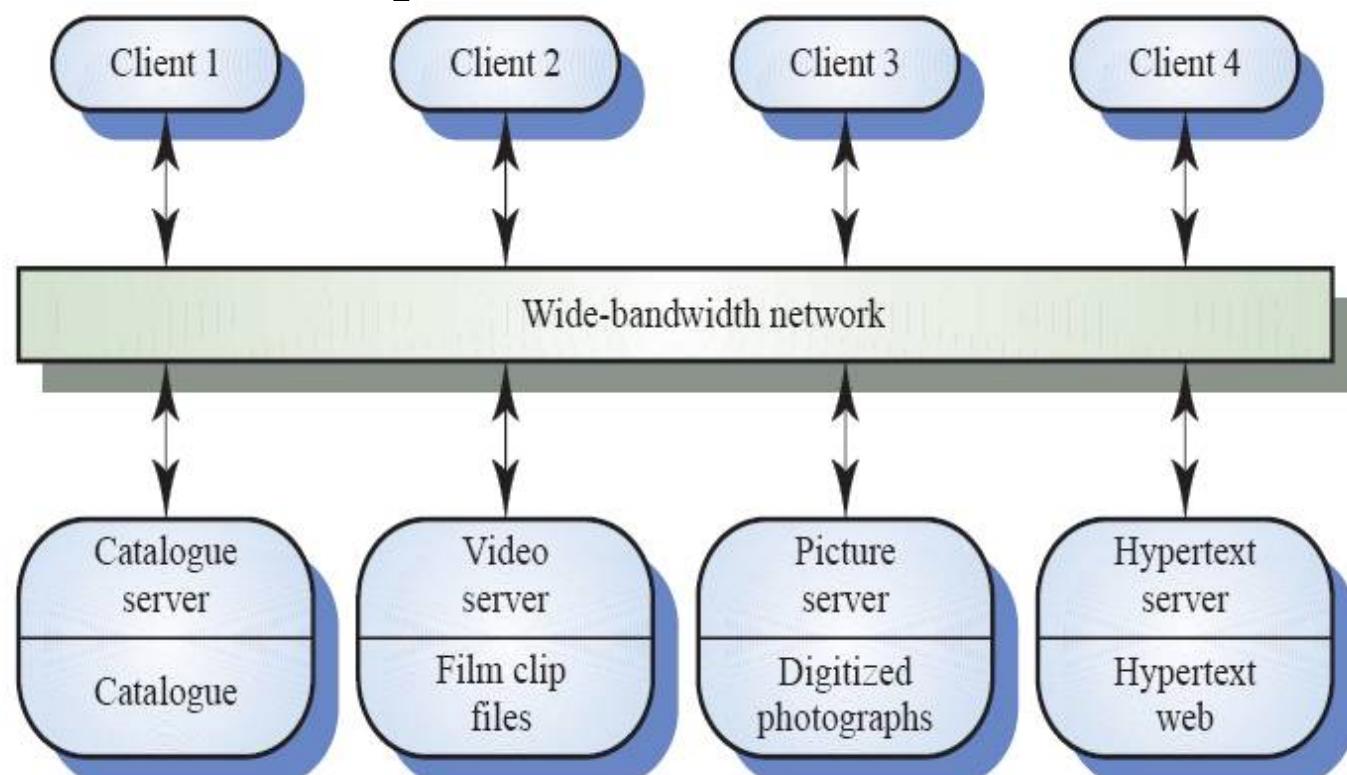
MÔ HÌNH KIẾN TRÚC- Mô hình cấu trúc

Kiến trúc khách-dịch vụ (Client–server architecture)

- Chỉ ra cách phân tán dữ liệu và xử lý thông qua quan hệ các thành phần
- Bao gồm tập hợp các máy chủ (server) độc lập cung cấp các dịch vụ đặc biệt như: in ấn, quản lý dữ liệu.
- Bao gồm tập hợp các người dùng (client) cần các dịch vụ
- Bao gồm hệ thống mạng cho phép các client truy cập các dịch vụ từ server

MÔ HÌNH KIẾN TRÚC- Mô hình cấu trúc

- Ví dụ kiến trúc khách-dịch vụ: Film & picture library



MÔ HÌNH KIẾN TRÚC- Mô hình cấu trúc

Kiến trúc khách-dịch vụ

• Ưu điểm

- Sử dụng hiệu quả mạng, không đòi hỏi thiết bị đắt tiền
- Dễ dàng mở rộng, thêm dịch vụ

• Nhược điểm

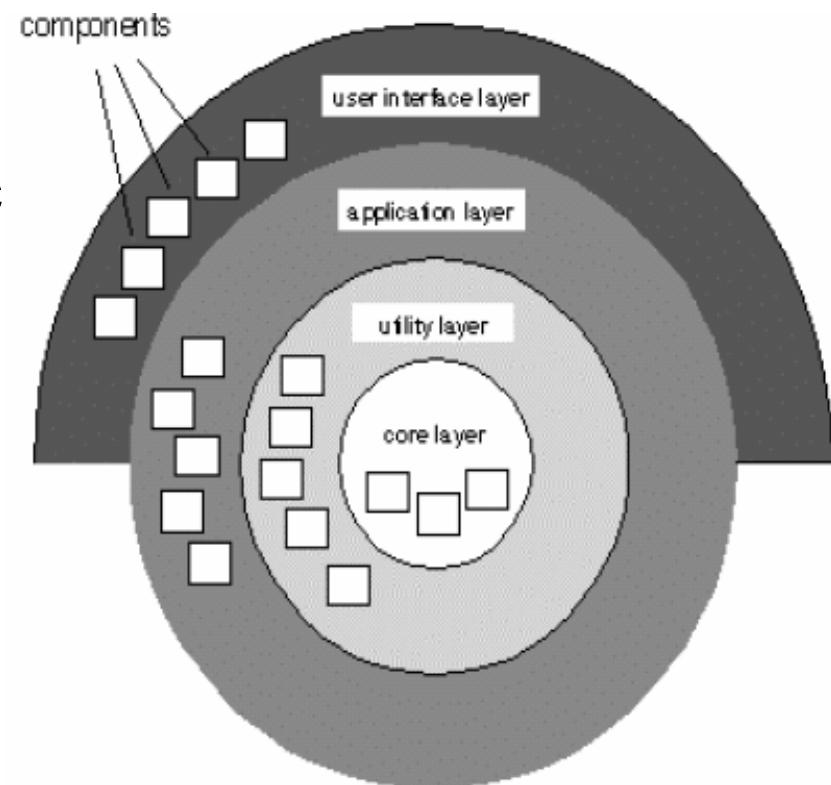
- Các hệ con dùng cấu trúc dữ liệu khác nhau, khó tích hợp
- Đòi hỏi cơ chế bảo toàn dữ liệu cho từng server

➡ Đang là mô hình phát triển ứng dụng phổ biến

MÔ HÌNH KIẾN TRÚC - Mô hình cấu trúc

Kiến trúc phân tầng - Layered Architecture (Abstract machine model)

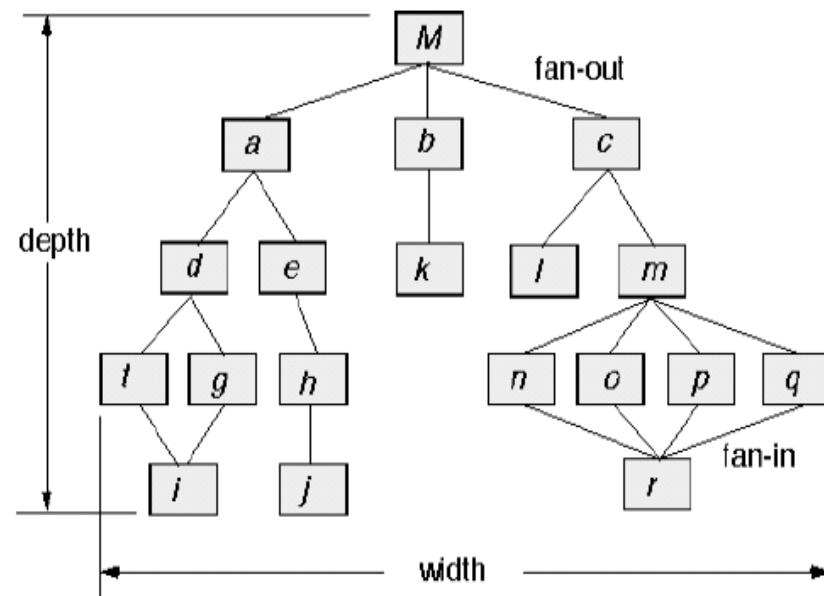
- Dùng để mô hình hóa giao diện của các phân hệ
- Phân rã hệ thống thành các tầng, mỗi tầng là một tập các dịch vụ
- Hỗ trợ sự phát triển tăng trưởng của các tầng, khi giao diện mỗi tầng thay đổi thì chỉ ảnh hưởng tới các tầng liền kề
- Không phải hệ thống nào cũng dễ dàng phân chia theo mô hình này



MÔ HÌNH KIẾN TRÚC - Mô hình điều khiển

Kiến trúc gọi và trả lại - Call and Return Architecture

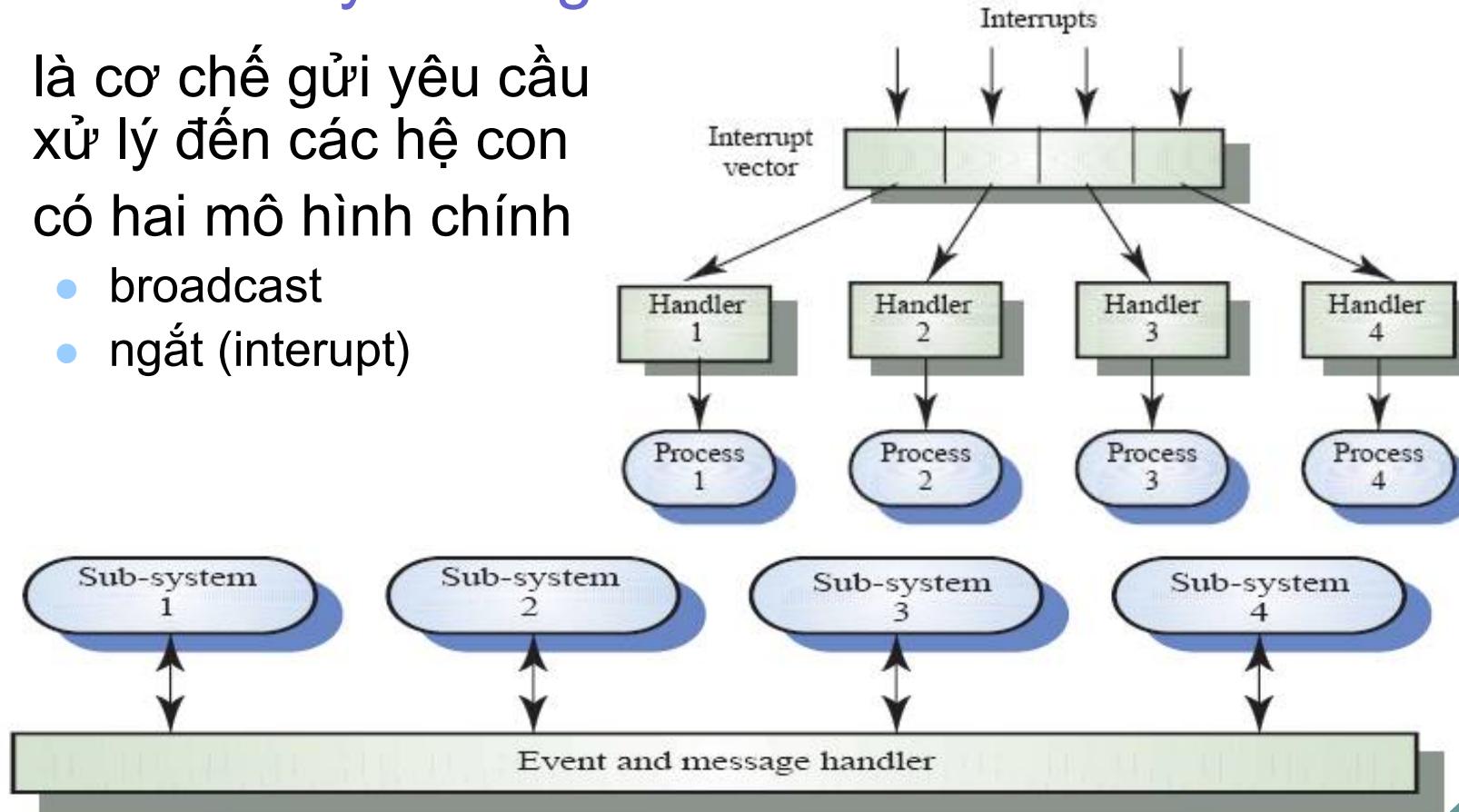
- Sử dụng một hệ con để điều khiển, khởi động và dừng các hệ thống con khác
- Truyền điều khiển trên xuống
- Dùng cho các hệ thống tuần tự



MÔ HÌNH KIẾN TRÚC - Mô hình điều khiển

Kiến trúc xử lý hướng sự kiện - Event-driven Architecture

- là cơ chế gửi yêu cầu xử lý đến các hệ con
- có hai mô hình chính
 - broadcast
 - ngắt (interrupt)



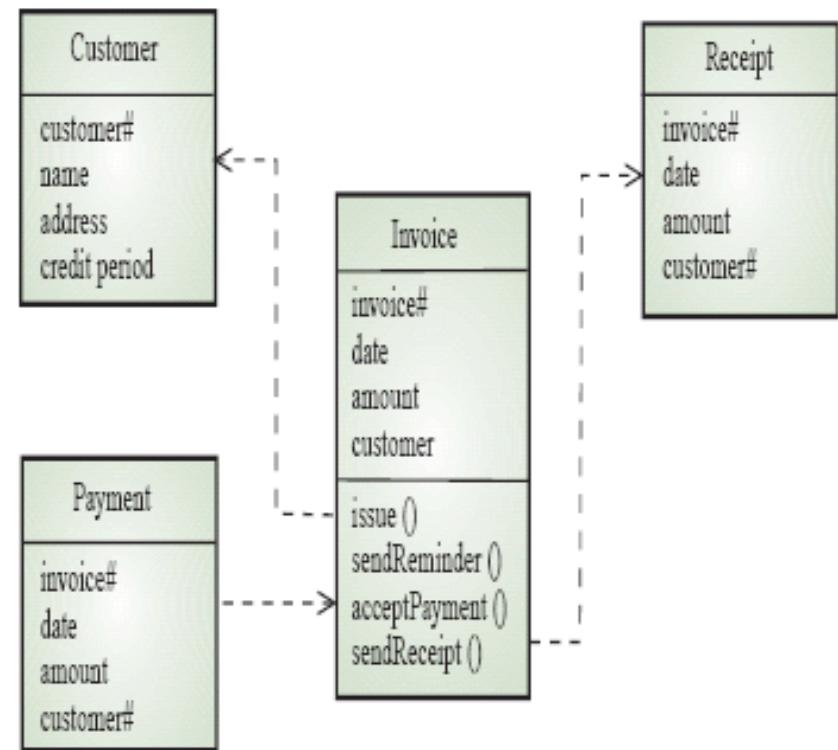
MÔ HÌNH KIẾN TRÚC - Phân rã mô đun

- Là cấp độ kiến trúc nhỏ nhất, các hệ thống con được phân rã thành các mô đun
- Hai mô hình phân rã mô đun
 - Mô hình đối tượng: hệ thống được phân ra thành tương tác đối tượng
 - Mô hình luồng dữ liệu: hệ thống được phân tách thành các mô đun chức năng chuyển các đầu vào thành các đầu ra (mô hình đường ống - pipeline)

MÔ HÌNH KIẾN TRÚC - Phân rã mô đun

Mô hình đối tượng

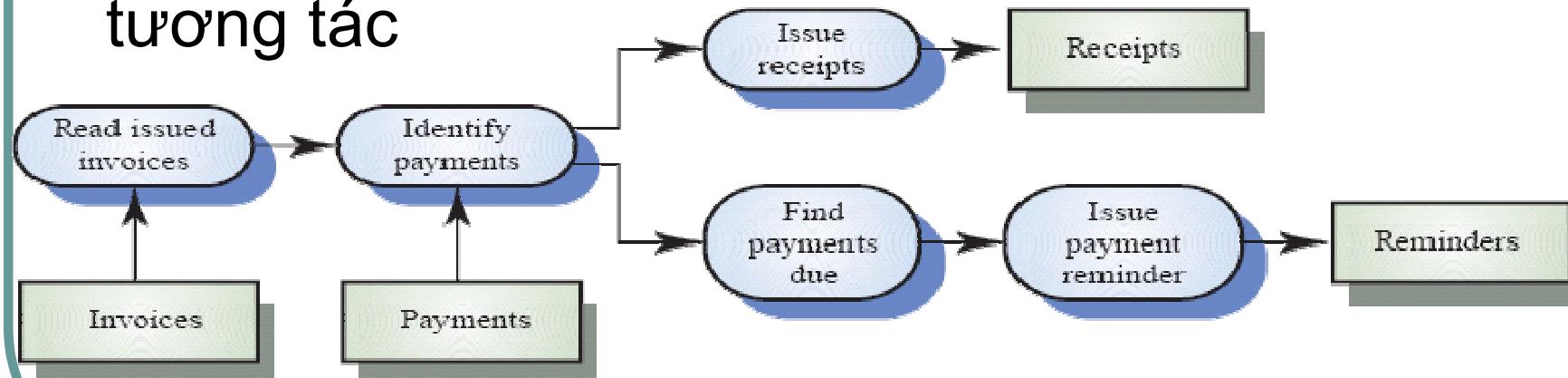
- Cấu trúc hệ thống thành tập hợp các đối tượng ghép nối lỏng lẻo qua giao diện
- Cần xác định: lớp đối tượng, thuộc tính, phương thức
- Khi triển khai: các đối tượng được tạo từ các lớp; một mô hình điều khiển được dùng để phối hợp thao tác đối tượng



MÔ HÌNH KIẾN TRÚC - Phân rã mô đun

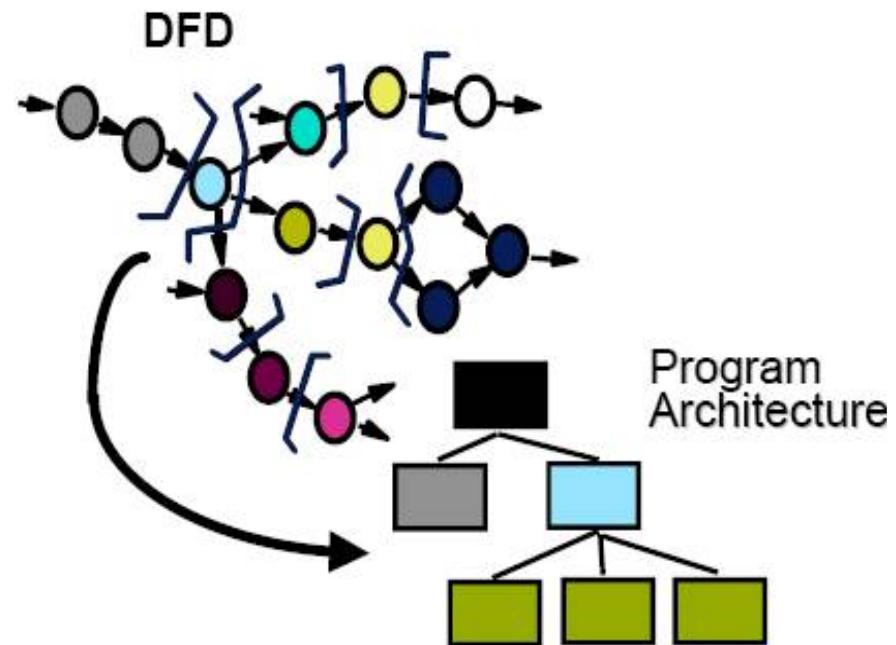
Mô hình luồng dữ liệu

- Tiến trình chuyển hóa (chức năng) các đầu vào thành các đầu ra
- Tương tự mô hình pipe, filter trong Unix
- Không thực sự phụ hợp cho các hệ thống tương tác



KỸ THUẬT THIẾT KẾ KIẾN TRÚC

- Mục tiêu: tạo ra kiến trúc được phân hoạch
- Cách tiếp cận: chuyển DFD thành kiến trúc phần mềm
- Ý nghĩa phân hoạch
 - Dễ kiểm thử, bảo trì
 - Hạn chế hiệu ứng phụ khi sửa đổi
 - Dễ mở rộng



KỸ THUẬT THIẾT KẾ KIẾN TRÚC

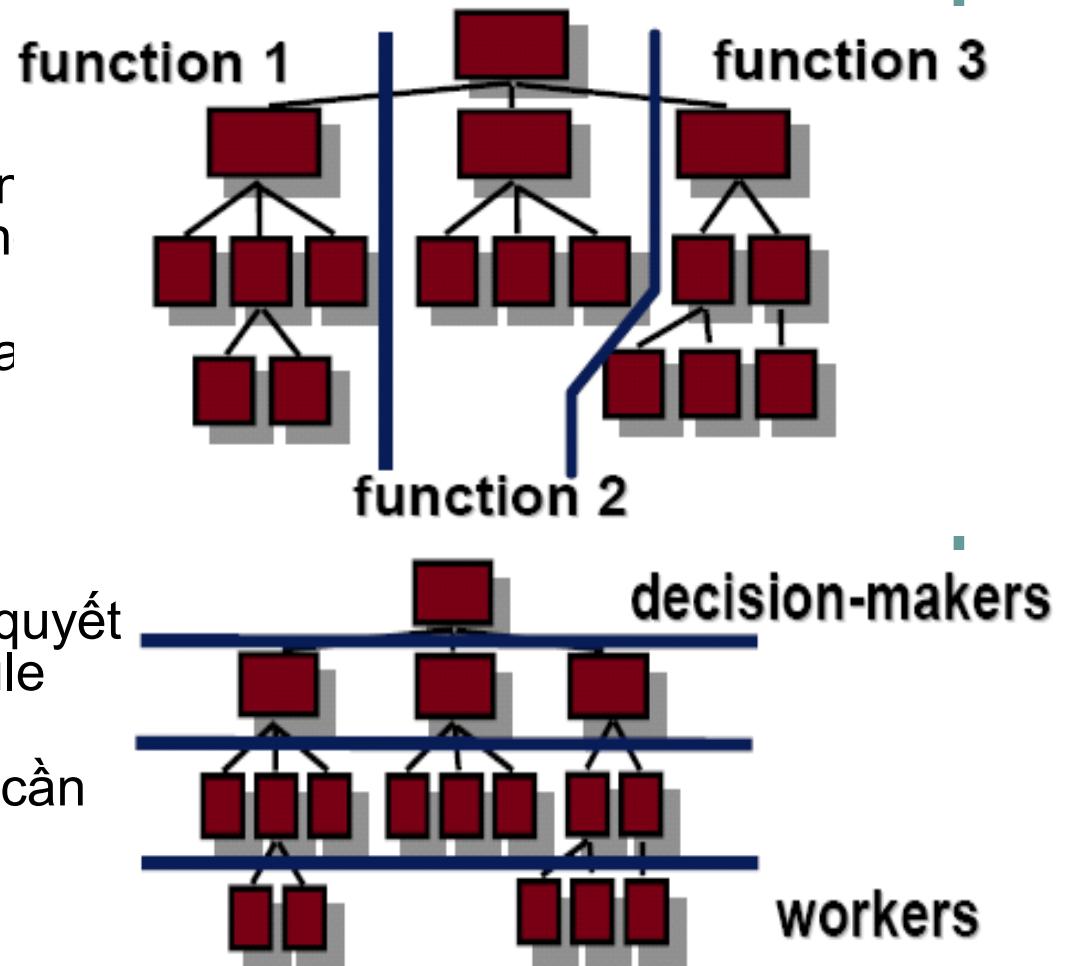
Phân hoạch kiến trúc

- Phân hoạch ngang

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

- Phân hoạch dọc

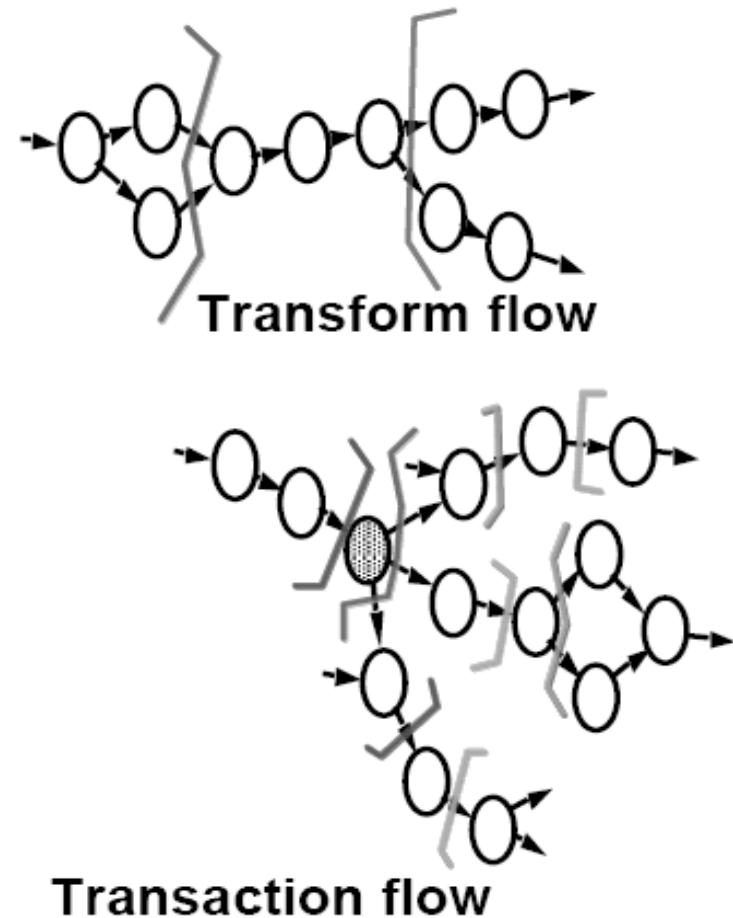
- phân tầng các module ra quyết định (điều khiển) và module thao tác
- các module ra quyết định cần được xếp ở tầng cao



KỸ THUẬT THIẾT KẾ KIẾN TRÚC

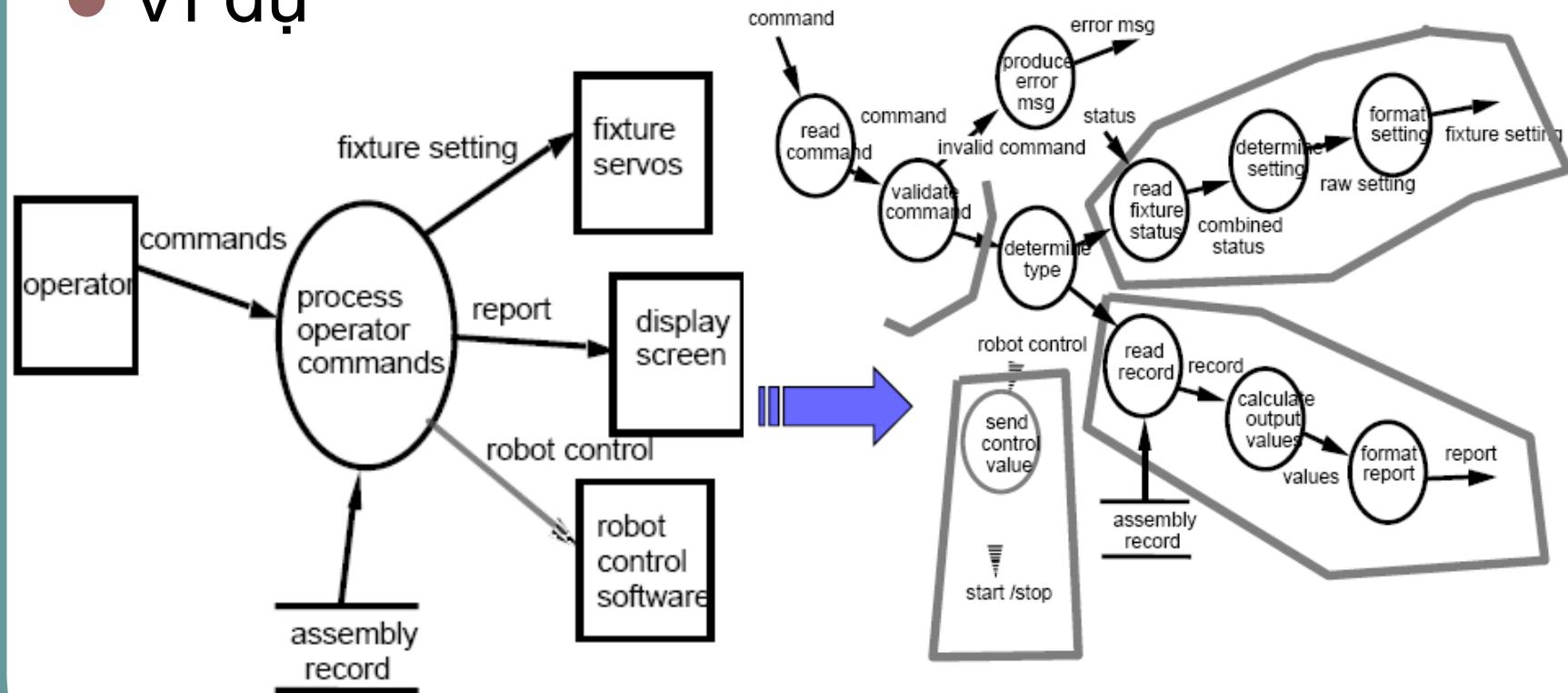
Phương pháp chuyển đổi

- Cô lập, xác định biên của các module vào/ra; xác định module giao tác (transaction)
- Chuyển đổi thành các module kiến trúc tương ứng
- Thêm các module điều khiển cần thiết
- Vi chỉnh (refining) kiến trúc để nâng cao tính module



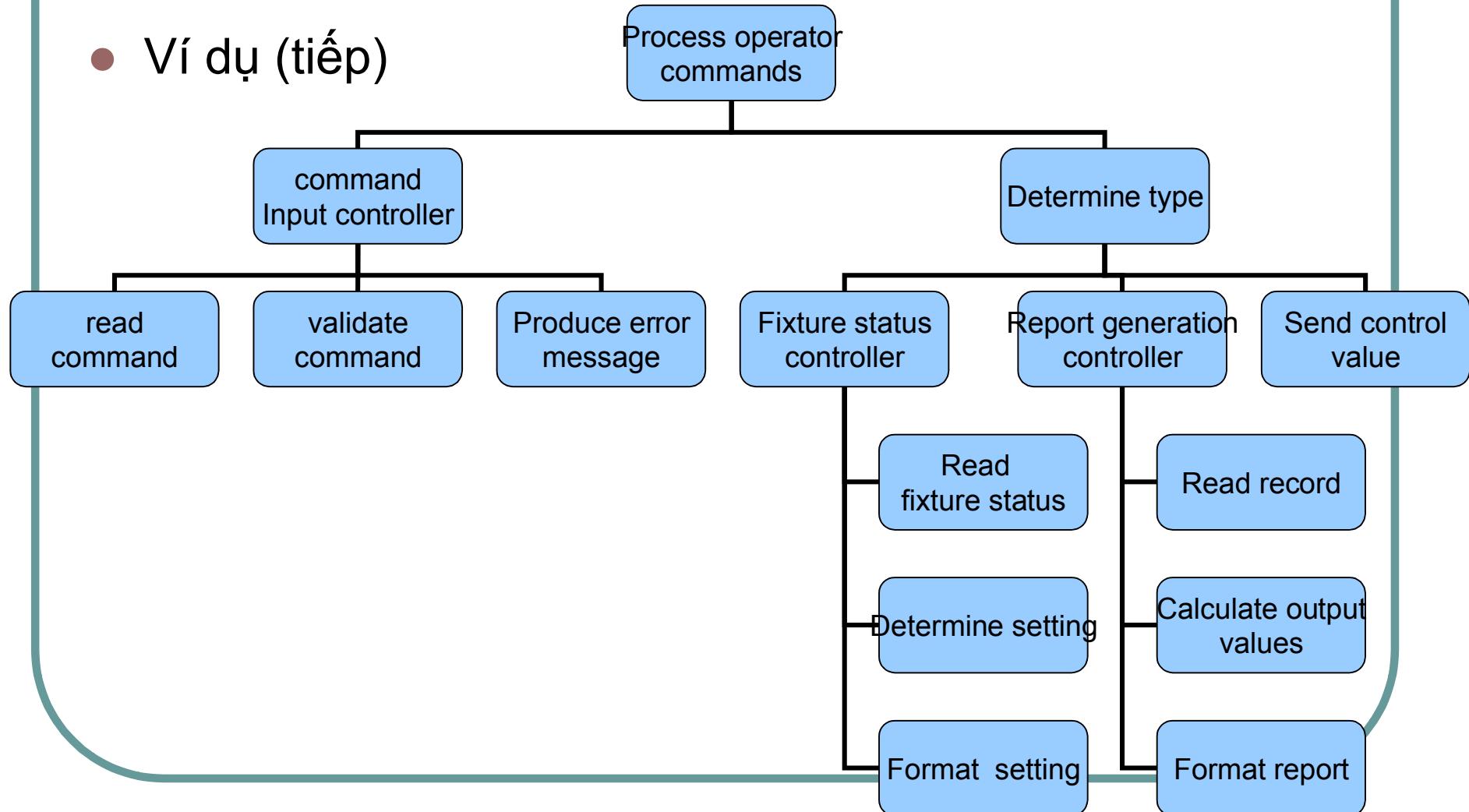
KỸ THUẬT THIẾT KẾ KIẾN TRÚC

● Ví dụ



KỸ THUẬT THIẾT KẾ KIẾN TRÚC

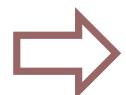
- Ví dụ (tiếp)



THIẾT KẾ GIAO DIỆN NGƯỜI DÙNG

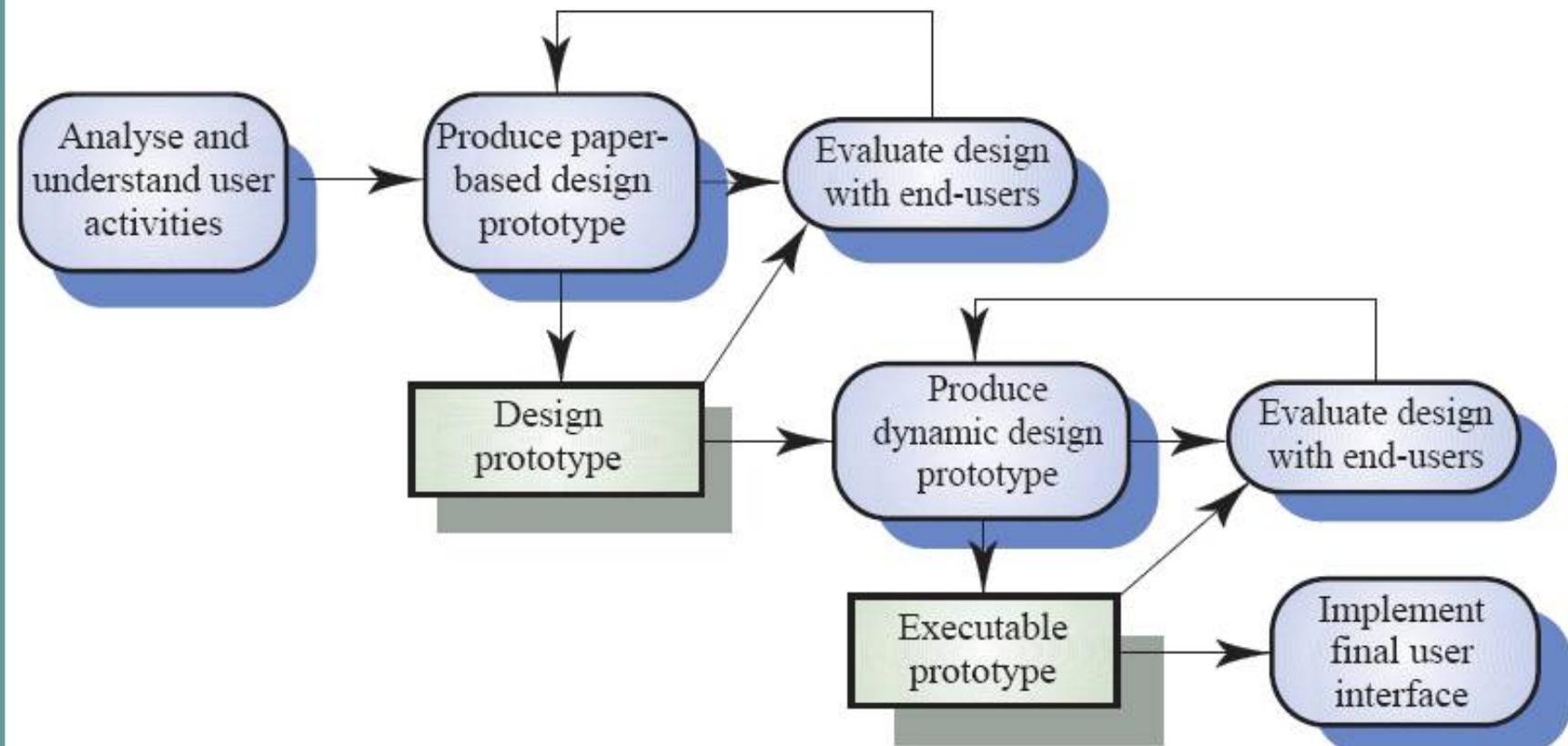
Thiết kế giao diện người dùng (User Interface)

- Là một khâu trong thiết kế phần mềm
 - Hướng người dùng
 - Làm bản mẫu, người dùng đánh giá
- Có vai trò quan trọng
 - Cho phép người dùng sử dụng hệ thống
 - Quyết định hiệu năng hệ thống
 - Người dùng đánh giá hệ thống qua giao diện



Cần dễ sử dụng + hiệu quả

THIẾT KẾ GDND – Tiến trình



THIẾT KẾ GDND – Nguyên lý

Yếu tố người dùng cần được xem xét trong thiết kế

- Nhu cầu của người dùng hệ thống
- Kinh nghiệm, năng lực
 - 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, biểu tượng

THIẾT KẾ GDND – Nguyên lý

Một số nguyên lý thiết kế giao diện người dùng

- Tính thân thiện người dùng
- Tính nhất quán
- Tính ít bất ngờ
- Tính phục lối
- Hướng dẫn người dùng
- Tính đa dạng về người dùng

THIẾT KẾ GDND – Kỹ thuật

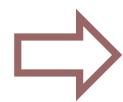
Hai vấn đề chính cần giải quyết

- **Tương tác người dùng:** cách người dùng đưa thông tin vào cho hệ thống
- **Biểu diễn thông tin:** cách hệ thống trình diễn thông tin cho người dùng
 - ➡ Giải pháp được xem xét theo góc độ
 - Thiết bị tương tác người dùng
 - Cách hệ thống trình diễn - chủng loại giao diện
 - Mô hình tương tác

THIẾT KẾ GDND – Thiết bị

Thiết bị vào và ra

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



Cả thiết bị lẫn phương thức đều đang tiến hóa - nhận dạng tiếng nói, chữ viết...

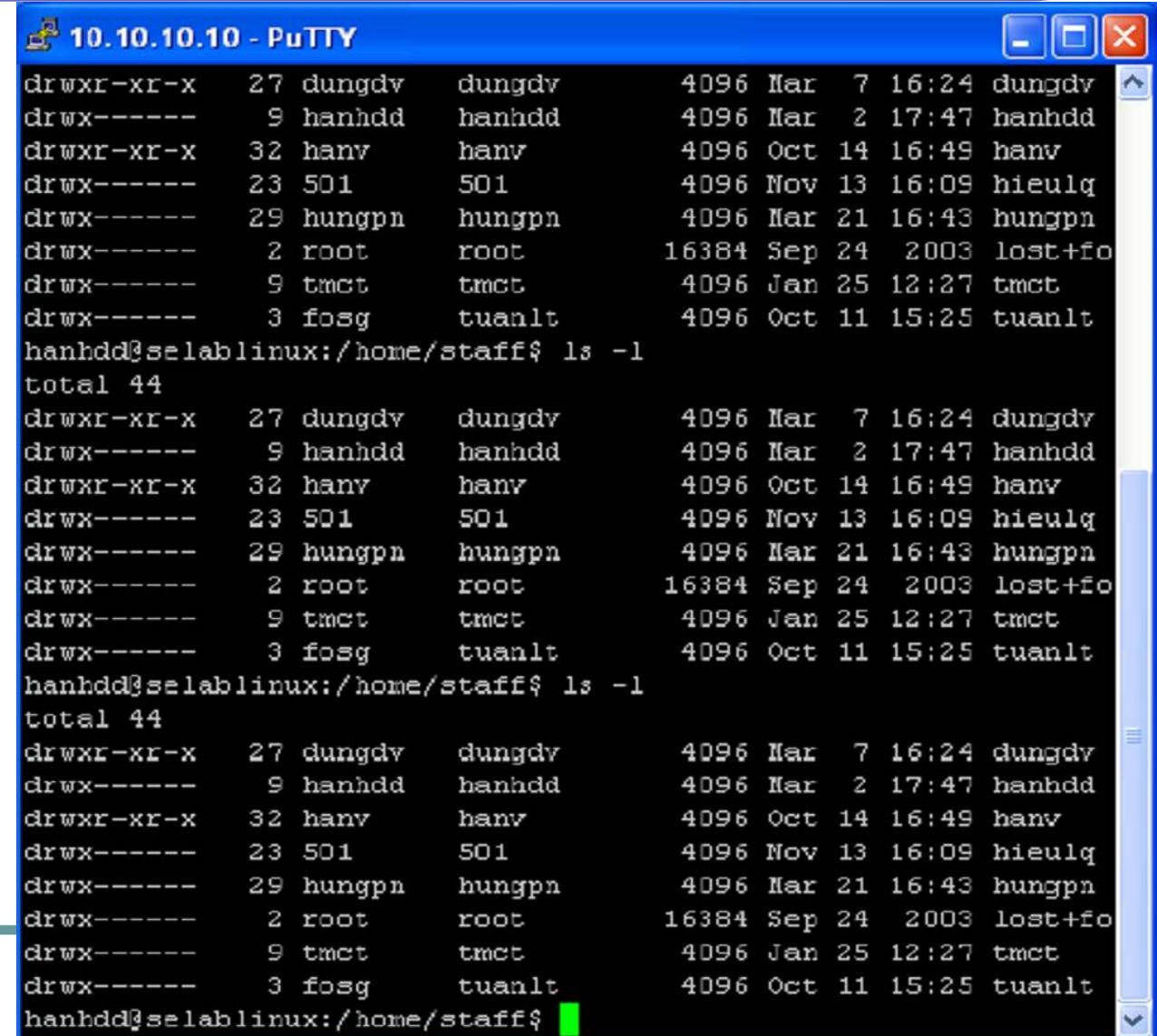
THIẾT KẾ GDND – Loại giao diện

Giao diện dòng lệnh

- Là phương thức tương tác đầu tiên
- 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
- 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

THIẾT KẾ GDND – Loại giao diện

Giao diện dòng lệnh



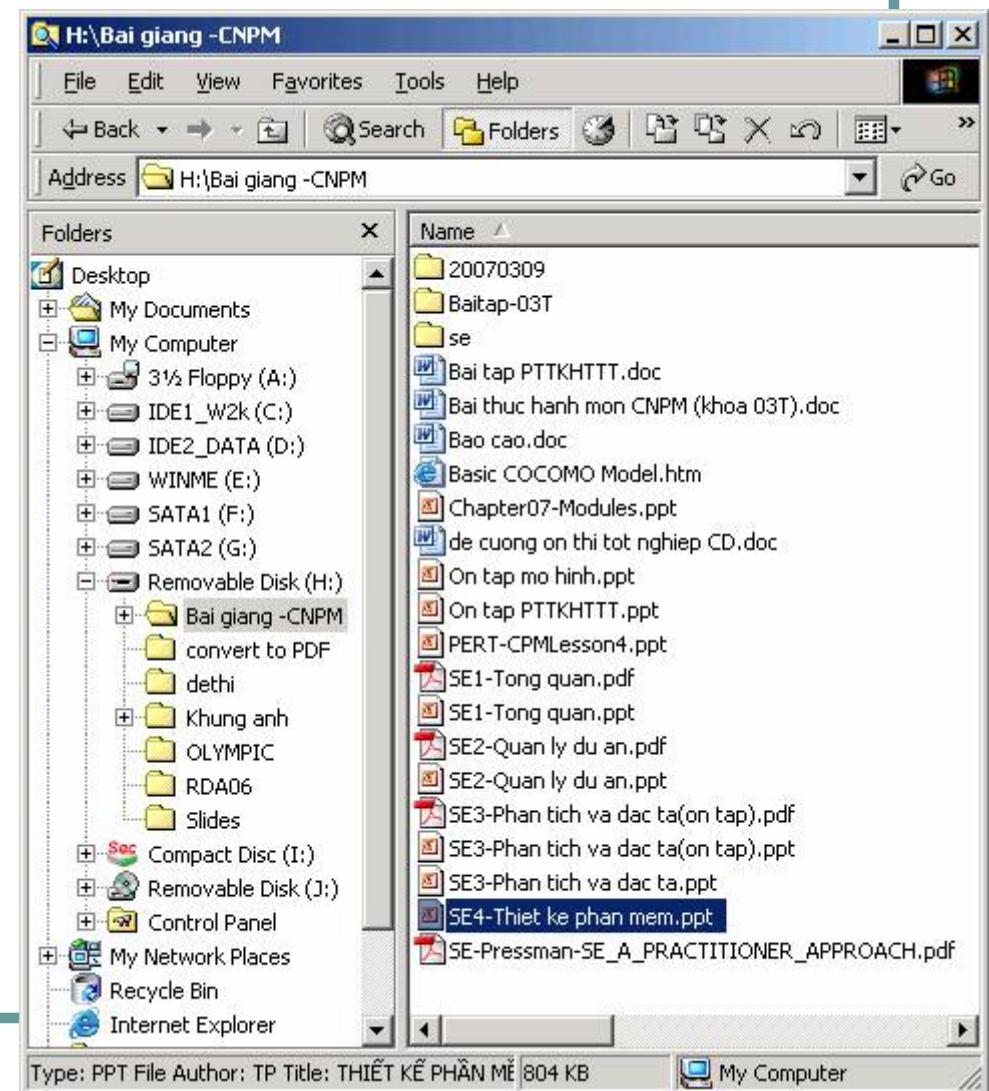
The screenshot shows a PuTTY terminal window titled "10.10.10.10 - PuTTY". The window displays a command-line interface where the user has run the "ls -l" command twice. The output lists files and directories in the "/home/staff" directory, showing details like permissions, owner, group, size, modification date, and time. The terminal window has a blue header bar and a white body with black text. It includes standard window controls (minimize, maximize, close) and scroll bars on the right side.

```
drwxr-xr-x  27 dungdv  dungdv        4096 Mar  7 16:24 dungdv
drwx-----  9 hanhdd  hanhdd        4096 Mar  2 17:47 hanhdd
drwxr-xr-x  32 hanv   hanv         4096 Oct 14 16:49 hanv
drwx----- 23 501    501          4096 Nov 13 16:09 hieulq
drwx----- 29 hungpn hungpn        4096 Mar 21 16:43 hungpn
drwx-----  2 root    root         16384 Sep 24 2003 lost+fo
drwx-----  9 tmct   tmct         4096 Jan 25 12:27 tmct
drwx-----  3 fosg   tuanlt       4096 Oct 11 15:25 tuanlt
hanhdd@selablinux:/home/staff$ ls -l
total 44
drwxr-xr-x  27 dungdv  dungdv        4096 Mar  7 16:24 dungdv
drwx-----  9 hanhdd  hanhdd        4096 Mar  2 17:47 hanhdd
drwxr-xr-x  32 hanv   hanv         4096 Oct 14 16:49 hanv
drwx----- 23 501    501          4096 Nov 13 16:09 hieulq
drwx----- 29 hungpn hungpn        4096 Mar 21 16:43 hungpn
drwx-----  2 root    root         16384 Sep 24 2003 lost+fo
drwx-----  9 tmct   tmct         4096 Jan 25 12:27 tmct
drwx-----  3 fosg   tuanlt       4096 Oct 11 15:25 tuanlt
hanhdd@selablinux:/home/staff$ ls -l
total 44
drwxr-xr-x  27 dungdv  dungdv        4096 Mar  7 16:24 dungdv
drwx-----  9 hanhdd  hanhdd        4096 Mar  2 17:47 hanhdd
drwxr-xr-x  32 hanv   hanv         4096 Oct 14 16:49 hanv
drwx----- 23 501    501          4096 Nov 13 16:09 hieulq
drwx----- 29 hungpn hungpn        4096 Mar 21 16:43 hungpn
drwx-----  2 root    root         16384 Sep 24 2003 lost+fo
drwx-----  9 tmct   tmct         4096 Jan 25 12:27 tmct
drwx-----  3 fosg   tuanlt       4096 Oct 11 15:25 tuanlt
hanhdd@selablinux:/home/staff$
```

THIẾT KẾ GDND – Loại giao diện

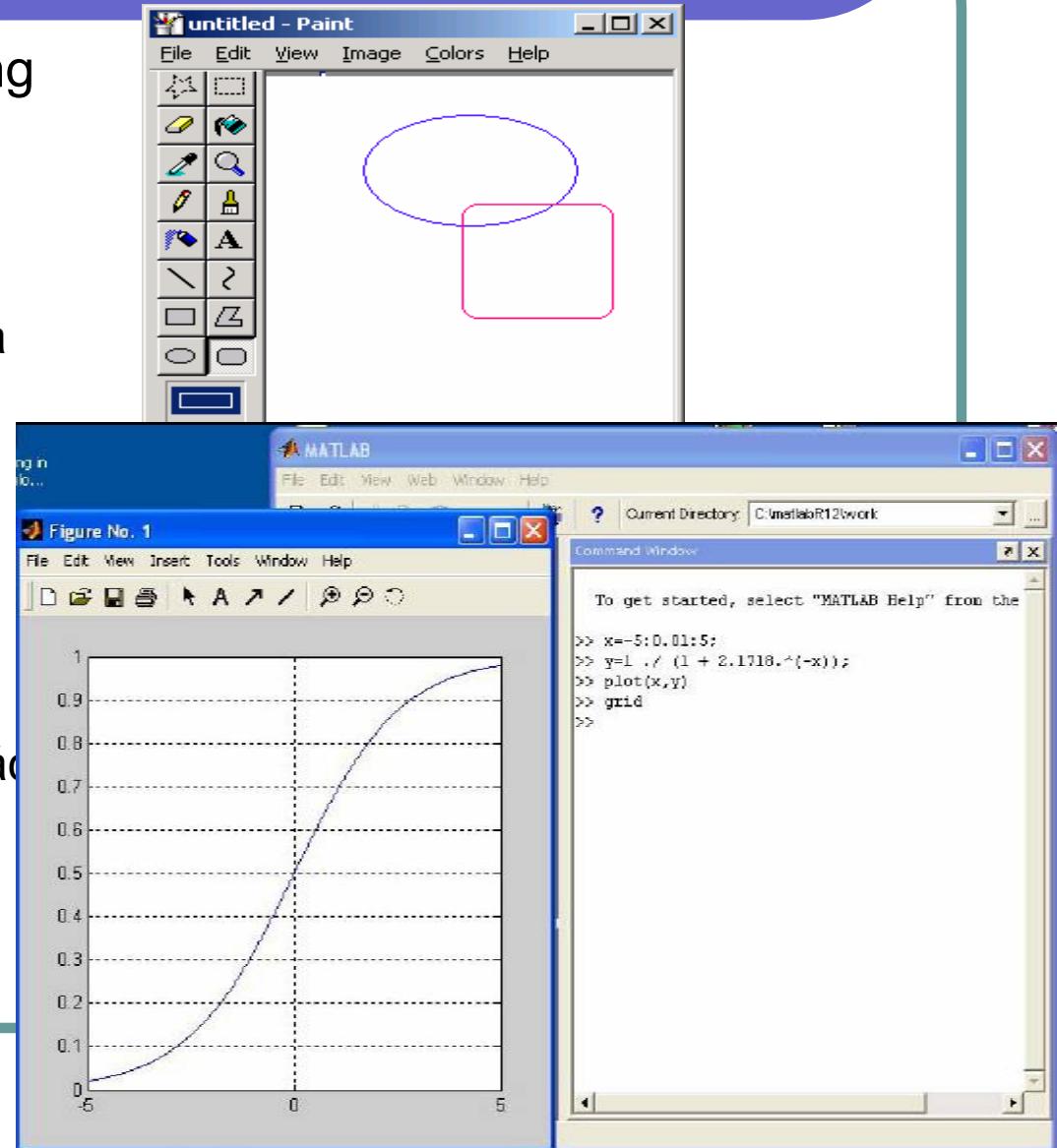
Giao diện đồ họa (GUI)

- Là giao diện 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 trên nhiều cửa sổ mà không bị mất thông tin
- Có thể hiển thị, tương tác dữ liệu trên nhiều vị trí trong cửa sổ



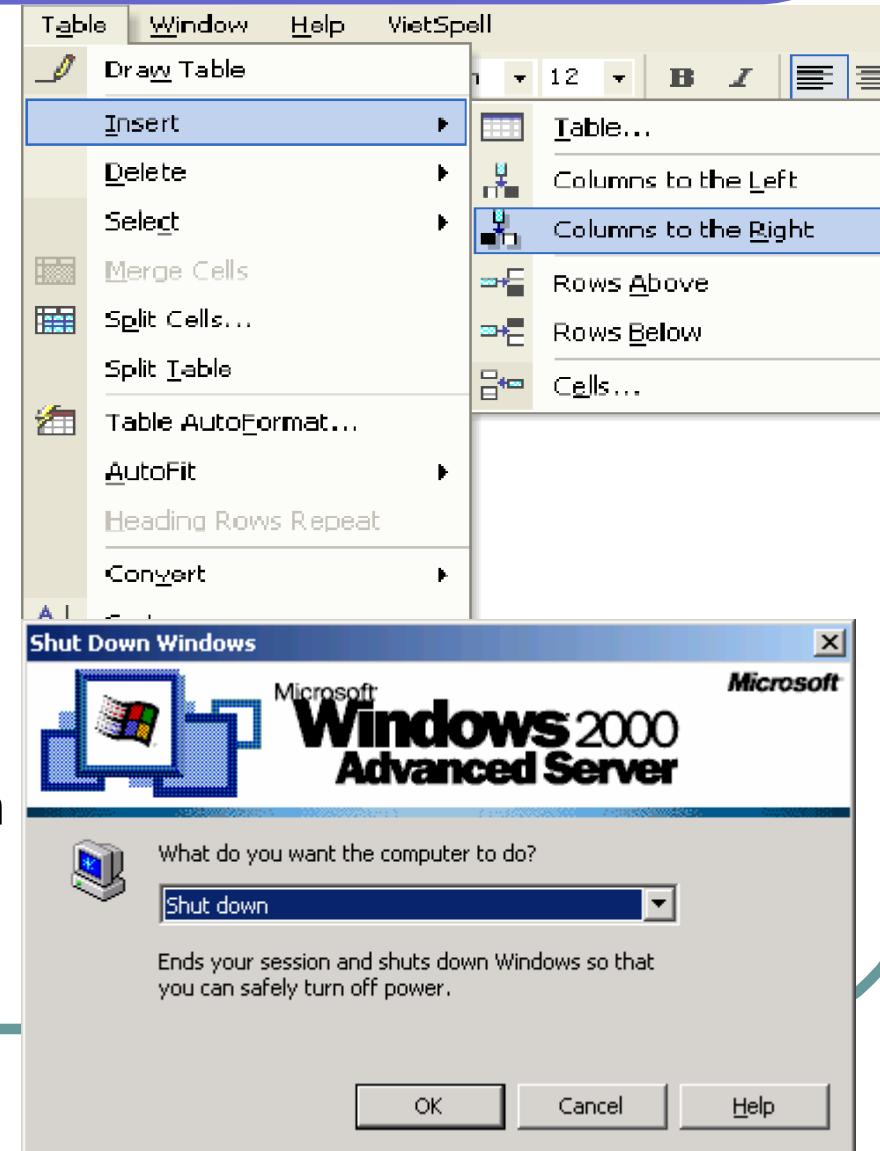
THIẾT KẾ GDND – Hình thức tương tác

- Tương tác trực tiếp với thông tin
 - ví dụ: 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
- 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



THIẾT KẾ GDND – Hình thức tương tác

- Sử dụng thực đơn (menu)
 - 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
- Đổi thoại (Dialog)
 - Dùng khi cần người dùng đưa ra lựa chọn quyết định

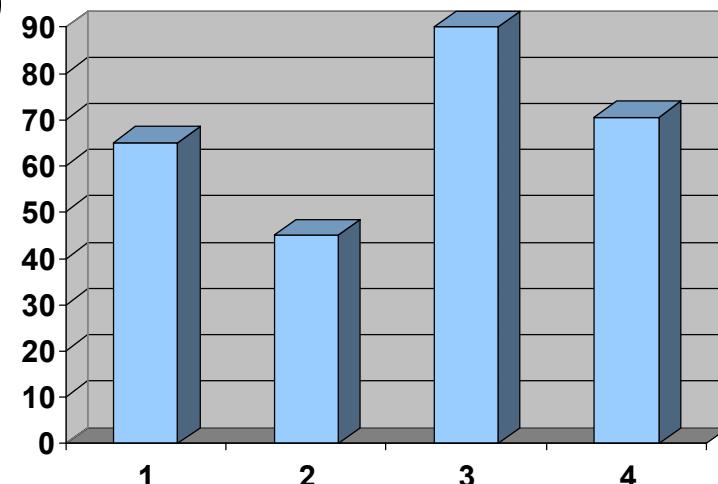


THIẾT KẾ GDND – Các vấn đề

Phương pháp hiện thị thông tin

- Hiển thị bằng văn bản (text)
 - chính xác
 - dễ cài đặt
- Hiển thị bằng đồ họa (graphic)
 - trực quan
 - dễ dàng nhìn nhận các mối quan hệ

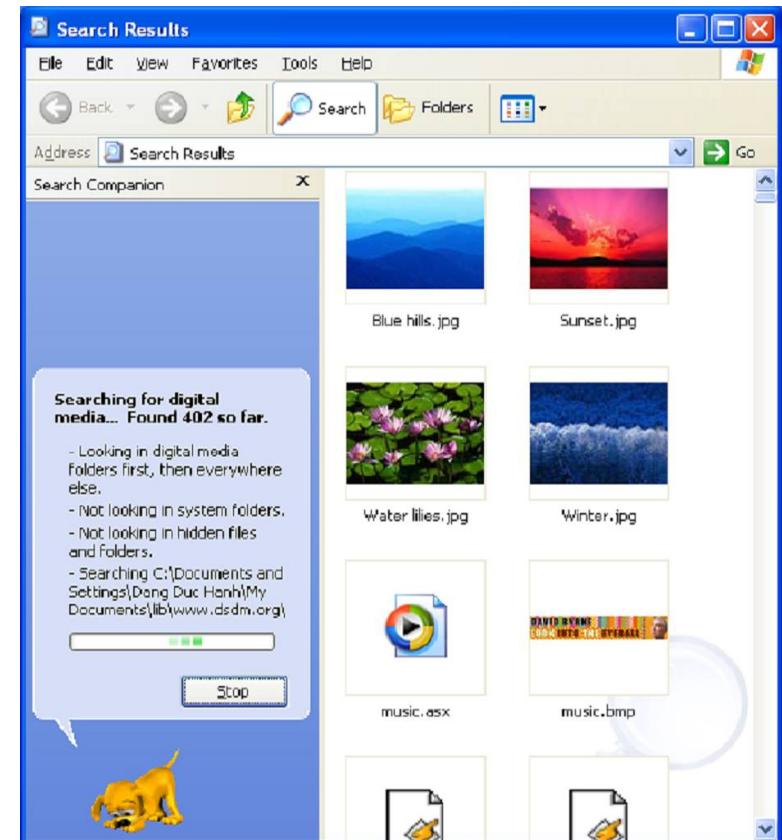
Tháng 1: 30
Tháng 2: 45
...



THIẾT KẾ GDND – Các vấn đề

Thời gian phản hồi

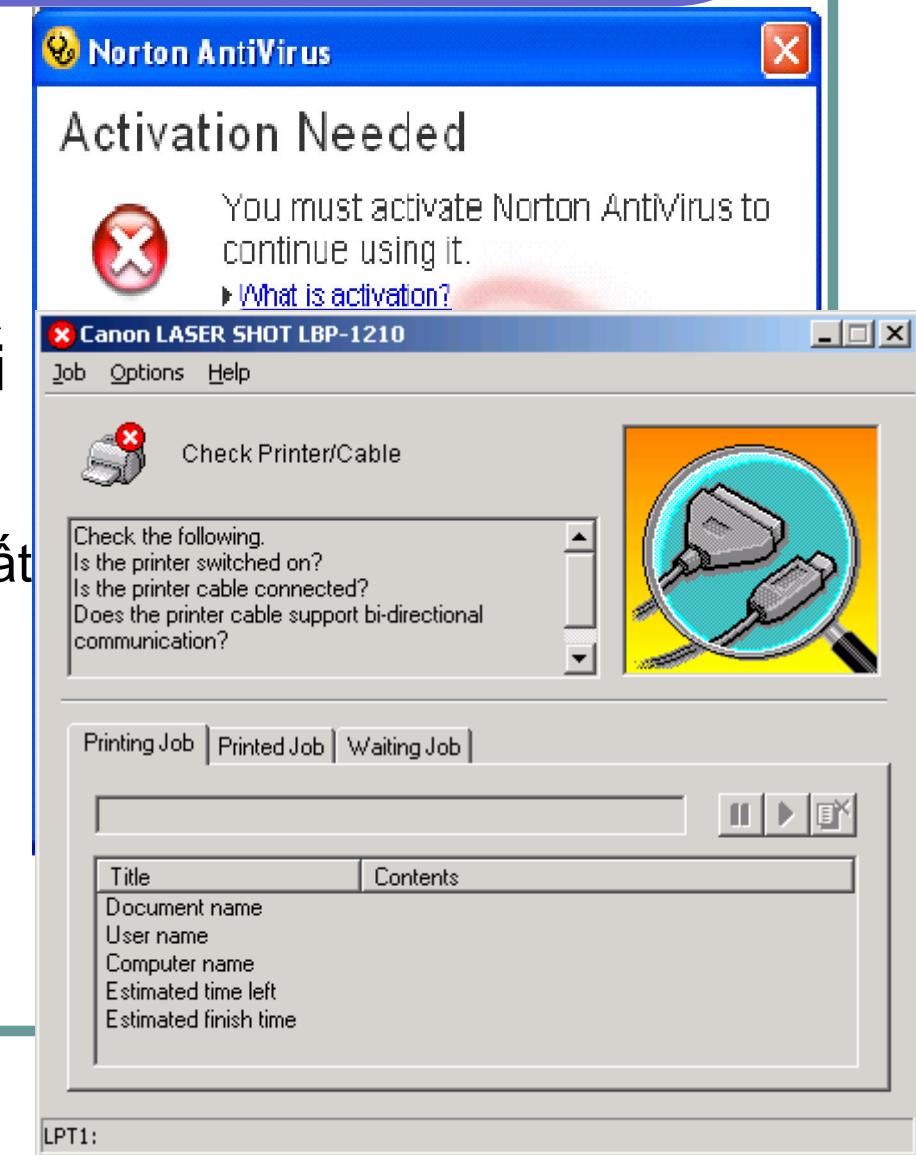
- Thời gian trung bình
 - thời gian trung bình phản hồi với thao tác
 - người dùng không thể đợi quá lâu (< 3s)
 - cần chứng tỏ hệ thống đang hoạt động
- Độ biến thiên thời gian phản hồi
- Gây cảm giác hệ thống gấp lối



THIẾT KẾ GDND – Các vấn đề

Thông báo

- Phản hồi của hệ thống đối với thao tác
- Cần có nghĩa, dễ hiểu, đưa ra các thông tin hữu ích với người dùng
 - tránh đưa ra các số hiệu
 - định dạng thông báo phải nhất quán
- Thông báo lỗi
 - chính xác
 - có tính xây dựng (nguyên nhân, cách khắc phục,...)



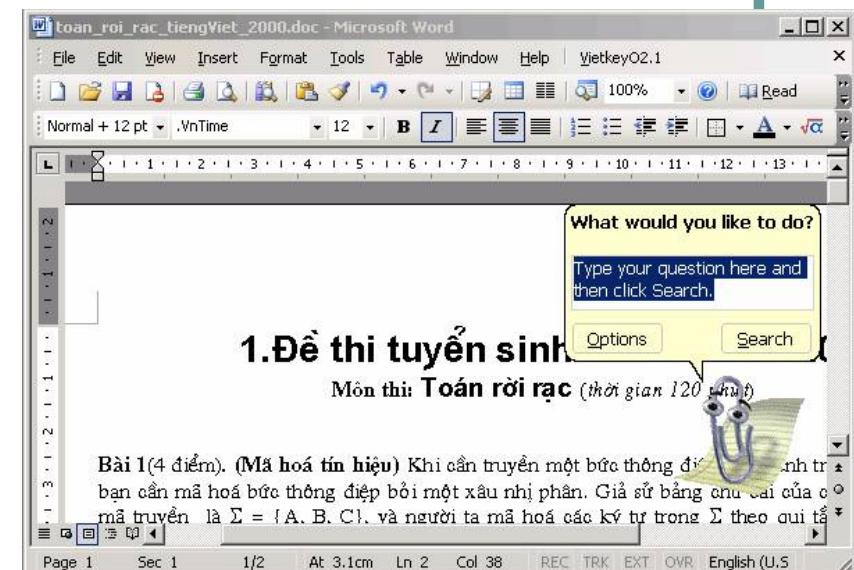
THIẾT KẾ GDND – Các vấn đề

Thông báo (2)

- 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 và thứ tự đưa ra thông báo
- Yêu cầu phản hồi đối với thông báo

THIẾT KẾ GDND – Các vấn đề

- Tiện ích (trợ giúp)
 - Cần có các tiện ích trợ giúp người sử dụng
- 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
 - ví dụ: MS Word macro



THIẾT KẾ GDND – Tính kỹ nghệ

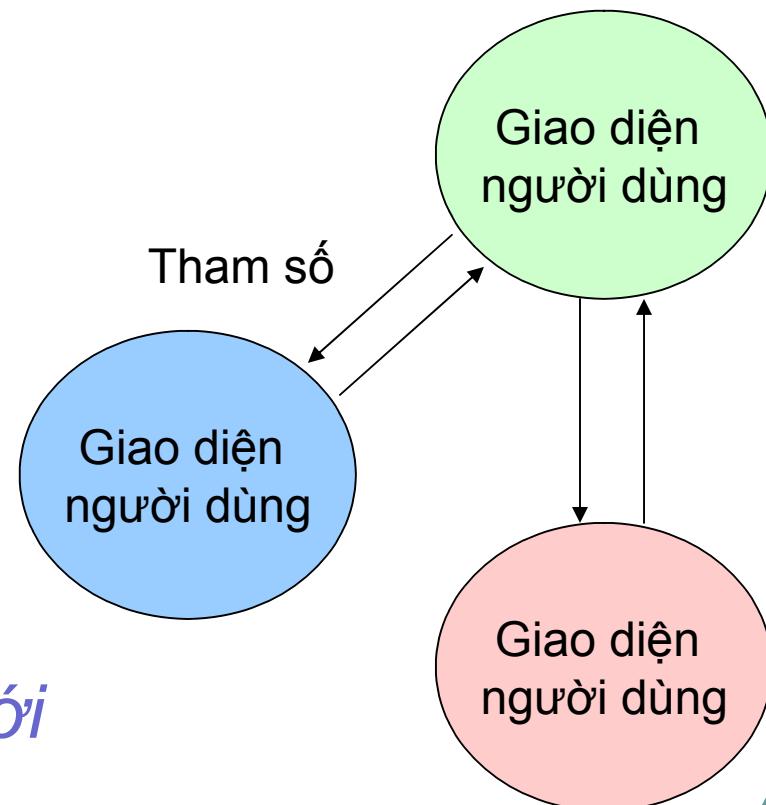
- Giao diện là phần tử dễ thay đổi

- thay đổi quy 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*



PHƯƠNG PHÁP CÔNG CỤ THIẾT KẾ

- Phương pháp thiết kế
 - Hướng chức năng
 - Hướng đối tượng
- Công cụ
 - Power Designer
 - Rational Rose

