

# ***Kỹ nghệ phần mềm***

## ***Software Engineering***

*Đại học Kinh doanh và Công nghệ Hà Nội*  
*Khoa CNTT*

*GV: Đào Thị Phụng*

*Email: [phuongdt102@gmail.com](mailto:phuongdt102@gmail.com)*

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

*Phone: 0946.866.817*

# Bài 5: Khái niệm thiết kế phần mềm



## Nội dung

- Khái niệm, nguyên lý, chất lượng
- Nội dung thiết kế và chất lượng

# 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.

# Khái niệm thiết kế phần mềm



- *Thiết kế là chuyển đặc tả yêu cầu thành mô tả thiết kế mà người lập trình có thể chuyển thành chương trình với 1 ngôn ngữ, vận hành được đáp ứng được yêu cầu đặt ra*
  - Là 1 quá trình sáng tạo:
    - ◆ Tìm giải pháp công nghệ (cách thức, phương án)
    - ◆ Biểu diễn cách thức, phương án
    - ◆ Xem xét lại, chi tiết hóa
- ➔ đủ chi tiết để người lập trình biết phải làm như thế nào để chuyển thành chương trình

# Vai trò thiết kế



- tạo mô hình cài đặt của phần mềm
- là công cụ giao tiếp giữa các những người tham gia phát triển, cơ sở đảm bảo chất lượng hệ thống
  - ◆ dễ đọc, dễ hiểu, dễ sửa đổi hơn mã chương trình
  - ◆ có nhiều mức chi tiết; cung cấp cái nhìn tổng thể
  - ◆ làm cơ sở để trao đổi, cải tiến
- Cung cấp đầy đủ thông tin cho việc bảo trì sau này:
  - Giảm công sức mã hóa khi sửa đổi
  - Tiện bảo trì phát triển, mở rộng

# Cấu trúc thiết kế



- Phần mềm là tập các mô đun tương tác lẫn nhau
- **Mô đun hóa** là chìa khóa cho phần mềm tốt
- Mục tiêu thiết kế là xác định:
  - ◆ các mô đun chức năng
  - ◆ cách thức cài đặt mô đun
  - ◆ tương tác giữa các mô đun

# Nguyên lý thiết kế



1. không bị bó buộc vào một cách nhìn hạn chế nào
  - ◆ nó cần được lựa chọn từ các giải pháp có thể
2. cho phép lần ngược lại mô hình phân tích
  - ◆ các mô đun & các yêu cầu không nhất thiết phải tương ứng 1-1
  - ◆ nhưng phải kiểm tra được sự thỏa mãn các yêu cầu

# Nguyên lý thiết kế (t)



3. Không nên tạo lại các thiết kế (giải pháp) đã có, mà cần tái sử dụng tối đa chúng
4. Mô hình thiết kế (*giải pháp*) nên tiến gần đến mô hình thế giới thực (*bài toán*)
5. Biểu diễn thiết kế phải **nhất quán** và có **tính tích hợp**:
  - ◆ thiết kế do nhiều người tiến hành song song
  - ◆ phải thống nhất cách biểu diễn, thống nhất giao diện
6. Thiết kế **cần có cấu trúc** để dễ hiểu, dễ thay đổi
  - ◆ phải được modun hóa, phân cấp



# Nguyên lý thiết kế (t)



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

# Nội dung & chất lượng thiết kế

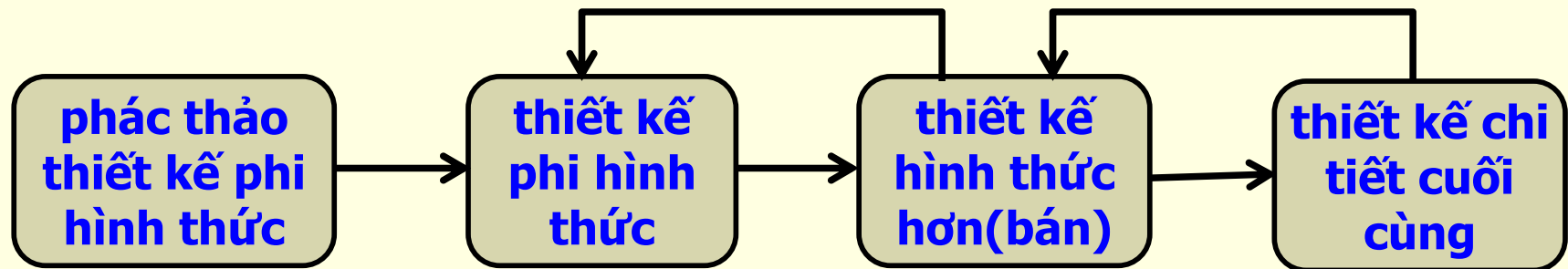


## Nội dung thiết kế

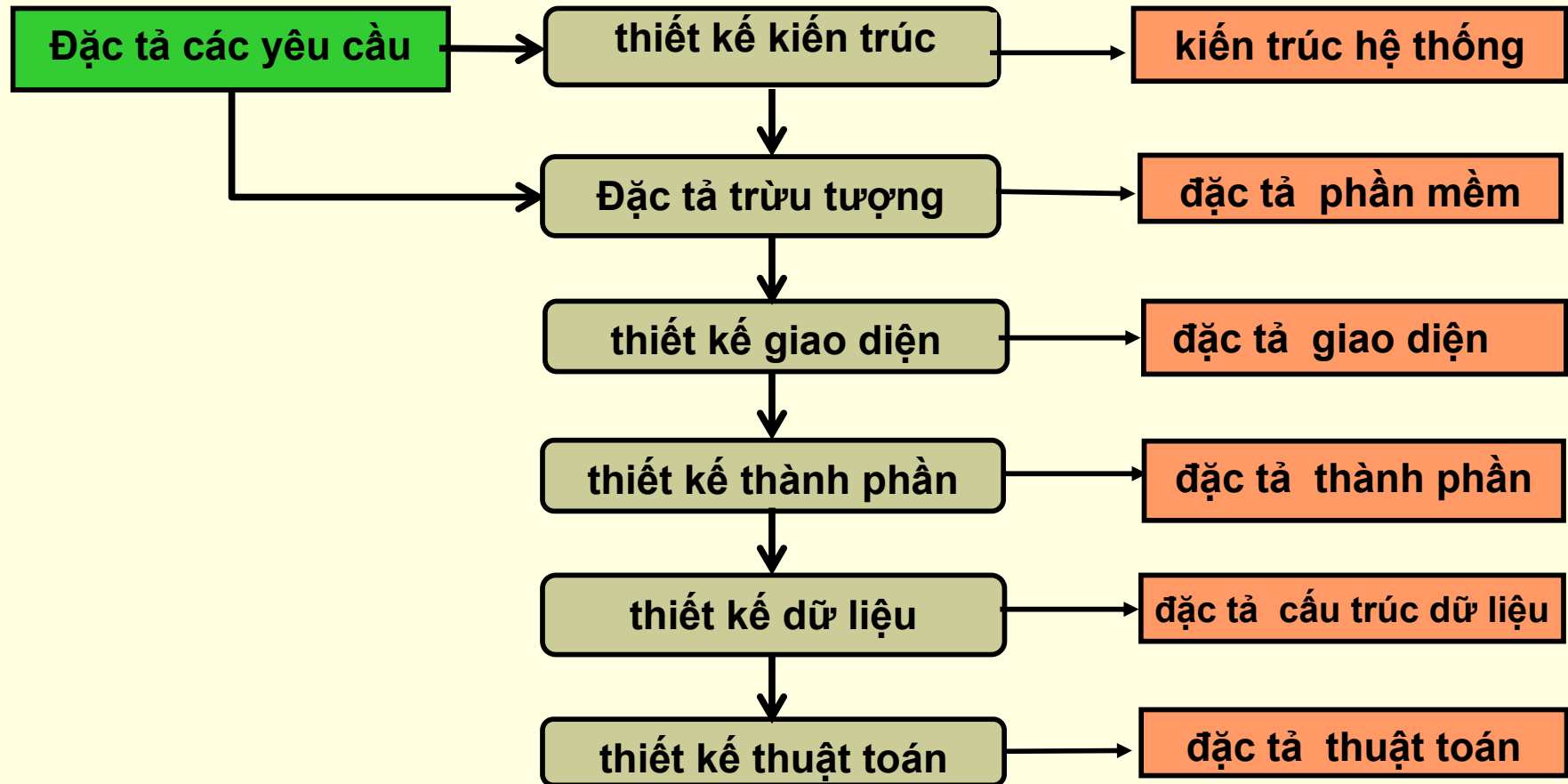
- Thiết kế kiến trúc
  - ◆ phân rã hệ thống thành hệ thống con các mô đun,
  - ◆ xác định giao diện tương tác giữa các mô đun
- Thiết kế cấu trúc dữ liệu
  - ◆ xây dựng mô hình biểu diễn thông tin
- Thiết kế thủ tục (thuật toán)
  - ◆ xác định các bước thực hiện xử lý
- Thiết kế giao diện người dùng
  - ◆ nên nhìn nhận giao diện là một bài toán độc lập

# Mô hình tổng quát tiến trình thiết kế

- Tiến trình thiết kế là quá trình tăng cường hình thức hóa và luôn quay lại các thiết kế đúng đắn và ít hình thức hóa hơn trước đó để kiểm tra và hoàn chỉnh



# Tiến trình hoạt động thiết kế và sản phẩm



# Thiết kế kiến trúc



Sử dụng biểu đồ cấu trúc (structure chart), mô tả:

- ☐ cái nhìn tổng thể về hệ thống
- ☐ mối quan hệ giữa các mô đun
- ☐ giao diện giữa các mô đun

không cần chỉ ra:

- ☐ thứ tự thực hiện
- ☐ số lần thực hiện
- ☐ chi tiết thiết kế

# Thiết kế cấu trúc dữ liệu



- ☐ Chọn cách biểu diễn các đối tượng thiết kế có ảnh hưởng mạnh mẽ đến chất lượng phần mềm

Các mức thiết kế

- ☐ **Thiết kế cấu trúc lô gic**

- ☐ Các quan hệ chuẩn
- ☐ Các khóa
- ☐ Các tham chiếu
- ☐ Các cấu trúc thao tác dữ liệu

**Thiết kế cấu trúc vật lý**

- Các file
- Các kiểu
- Kích cỡ

# Thiết kế thủ tục



- ☐ Mô tả các bước hoạt động của mô đun
- ☐ Phương pháp mô tả
  - giả mã (pseudo code)
  - sơ đồ luồng (flow chart)
  - biểu đồ (diagram) Nassi-Shneiderman
  - biểu đồ hoạt động (activity diagram)
  - JSP

# Các khái niệm thiết kế cơ sở



- **Trừu tượng hóa:** trừu tượng hóa dữ liệu, thủ tục, điều khiển
- **Làm mịn:** chi tiết hóa các trừu tượng theo ý đồ
- **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

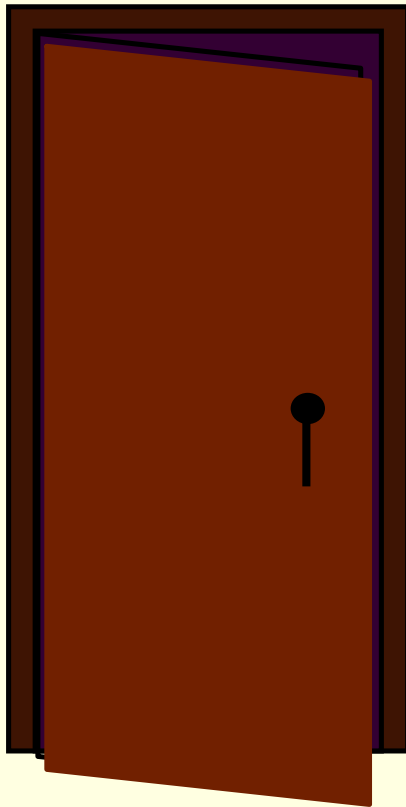


# Trừu tượng hóa (*abstraction*)



- 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 1 khái niệm logic
- Có nhiều mức trừu 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

# Trừ tượng dữ liệu



## Cửa

mã số: 256AD

loại: *cửa ra vào*

hướng mở: *ra bên trái*

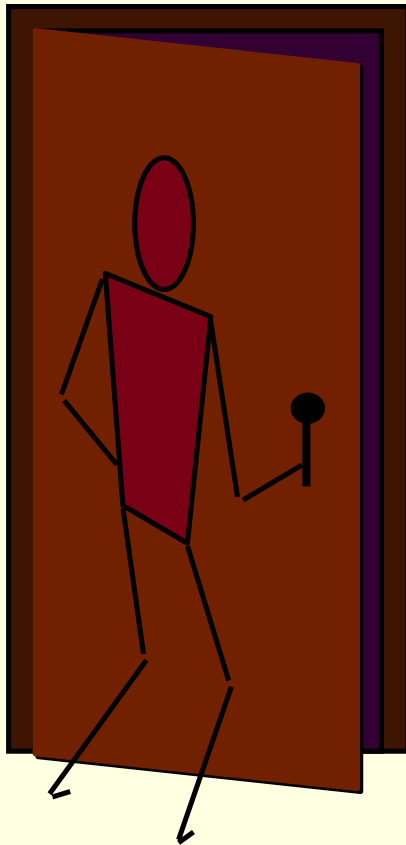
cao: 2.3

rộng: 0.85

trong lượng: 120

màu: *nâu cánh dán*

# Trừ tượng thủ tục



## Mở cửa

Mô tả chi tiết quá trình vào phòng qua cửa

# Làm mịn từng bước



## Mở cửa

Bước đến gần cửa  
Đưa chìa khóa vào ổ xoay

Mở cửa

Bước qua vào phòng  
Đóng cửa lại

Lặp lại cho đến khi chốt khóa bật ra  
Nếu chốt không mở thì  
Rút khóa ra, tìm chìa khác phù  
hợp, cắm vào ổ khóa, tiếp tục  
xoay cho đến khi mở được  
Bước qua cửa vào phòng  
Đóng cửa lại

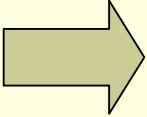
# Thiết kế mô đun



Dựa trên quan điểm "chia để trị"

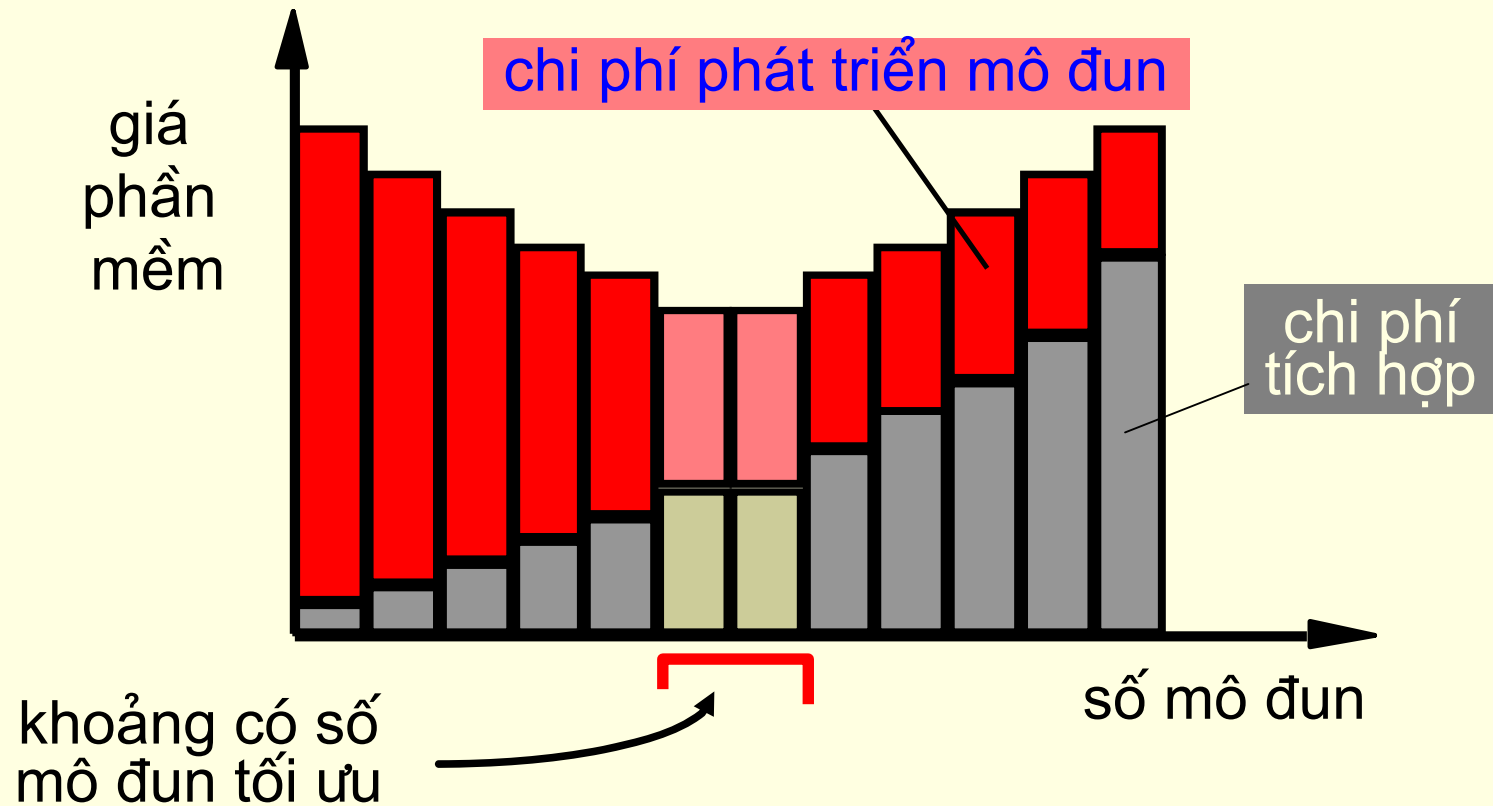
C: độ phức tạp  $C(p1 + p2) > C(p1) + C(p2)$

E: cùng sức thực hiện  $E(p1 + p2) > E(p1) + E(p2)$

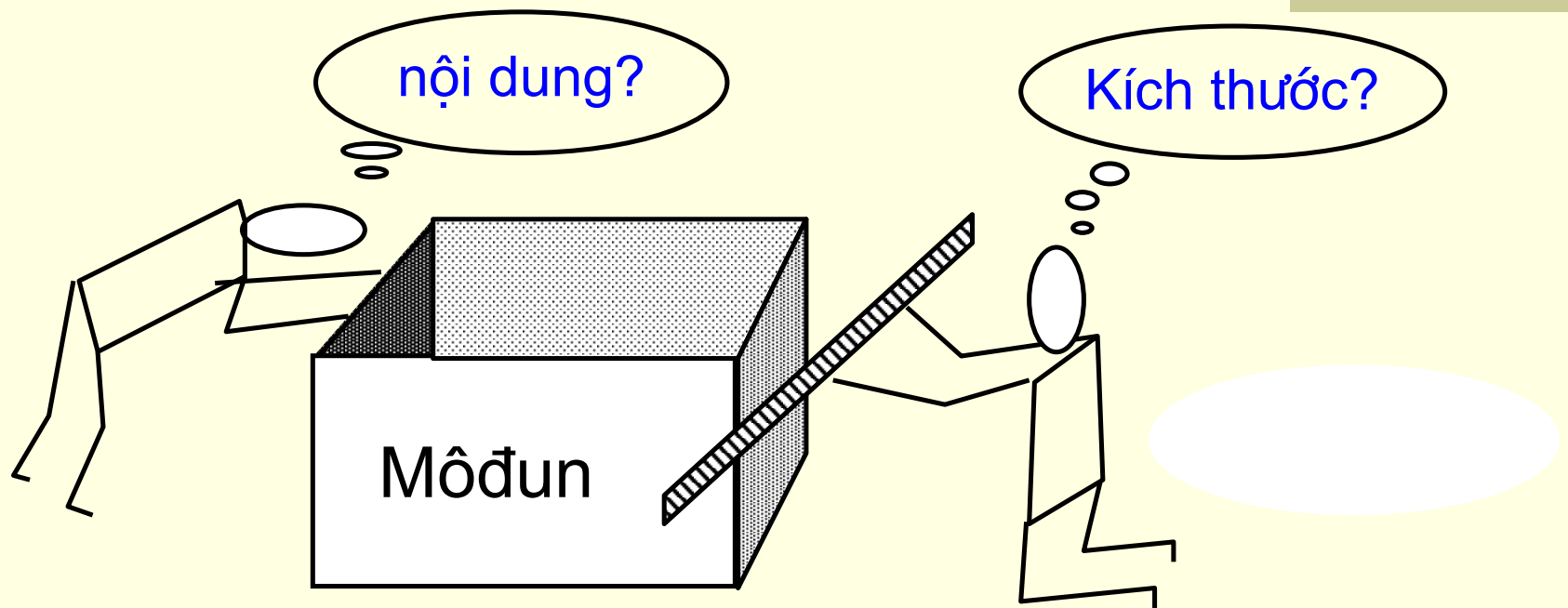
- 
- ☐ giảm độ phức tạp
  - ☐ cục bộ, dễ sửa đổi
  - ☐ có khả năng phát triển song song
  - ☐ dễ sửa đổi, dễ hiểu nên dễ tái sử dụng

# Số lượng mô đun

Cần xác định số mô đun tối ưu



# Kích cỡ mô đun



- Kích cỡ mô đun được quyết định dựa trên khái niệm độc lập chức năng: mỗi mô đun nên thực hiện 1 công việc: dễ hiểu, dễ sửa đổi, dễ tái sử dụng

# Che giấu 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 của nó:
  - thuật toán
  - cấu trúc dữ liệu
  - giao diện ngoại lai (mô đun thứ cấp, thiết bị vào/ra)
  - tài nguyên hệ thống



# Lý do che giấu thông tin



- Giảm hiệu ứng phụ khi sửa đổi mô đun
- Giảm tác động của thiết kế tổng thể lên thiết kế cục bộ
- Nhấn mạnh trao đổi thông tin thông qua giao diện
- Loại bỏ việc sử dụng dữ liệu dùng chung
- Hướng tới sự đóng gói chức năng, 1 thuộc tính của thiết kế tốt

Tạo ra các sản phẩm phần mềm tốt hơn

# Chất lượng thiết kế



**Ba đặc trưng** xem như là hướng dẫn cho 1 thiết kế tốt (McMlaughli[MCG91]):

- Thiết kế phải **triển khai được tất cả yêu cầu** trong mô hình phân tích & yêu cầu tiềm ẩn mà khách hàng đòi hỏi.
- Thiết kế cần **là bản hướng dẫn dễ đọc, dễ hiểu** cho người viết chương trình, người kiểm thử và người bảo trì.
- Thiết kế cần cung cấp **1 bức tranh đầy đủ về phần mềm trên quan điểm triển khai** hướng đến các mặt dữ liệu, chức năng và hành vi của hệ thống

**MCG91:** McLaughli,R., Một số chú ý về thiết kế chương trình, Software Engineering Notes,vol.16, no.4,oct 1991, pp53-54.

# Tiêu chí chất lượng



Cần thiết lập các tiêu chí kỹ thuật để đánh giá một thiết kế tốt hay không:

- Thiết kế cần **có kiến trúc tốt**: cấu thành từ các mẫu (pattern), các thành phần có đặc trưng tốt, dễ tiến hoá.
- Thiết kế được **môđul hoá** cho mỗi thành phần chức năng
- Chứa các **biểu diễn tách biệt** nhau về: dữ liệu, kiến trúc, giao diện, thành phần, môi trường
- **Liên kết qua giao diện** làm giảm độ phức tạp liên kết giữa các môđul với nhau và giữa hệ thống và môi trường

# Độ đo chất lượng thiết kế



- Phụ thuộc bài toán, không có phương pháp chung
- Một số độ đo:
  - ◆ **Coupling**: mức độ ghép nối giữa các module
  - ◆ **Cohesion**: mức độ liên kết giữa các thành phần trong một module
  - ◆ **Understandability**: tính hiểu được
  - ◆ **Adaptability**: tính thích nghi được

# Độ đo chất lượng thiết kế (t)



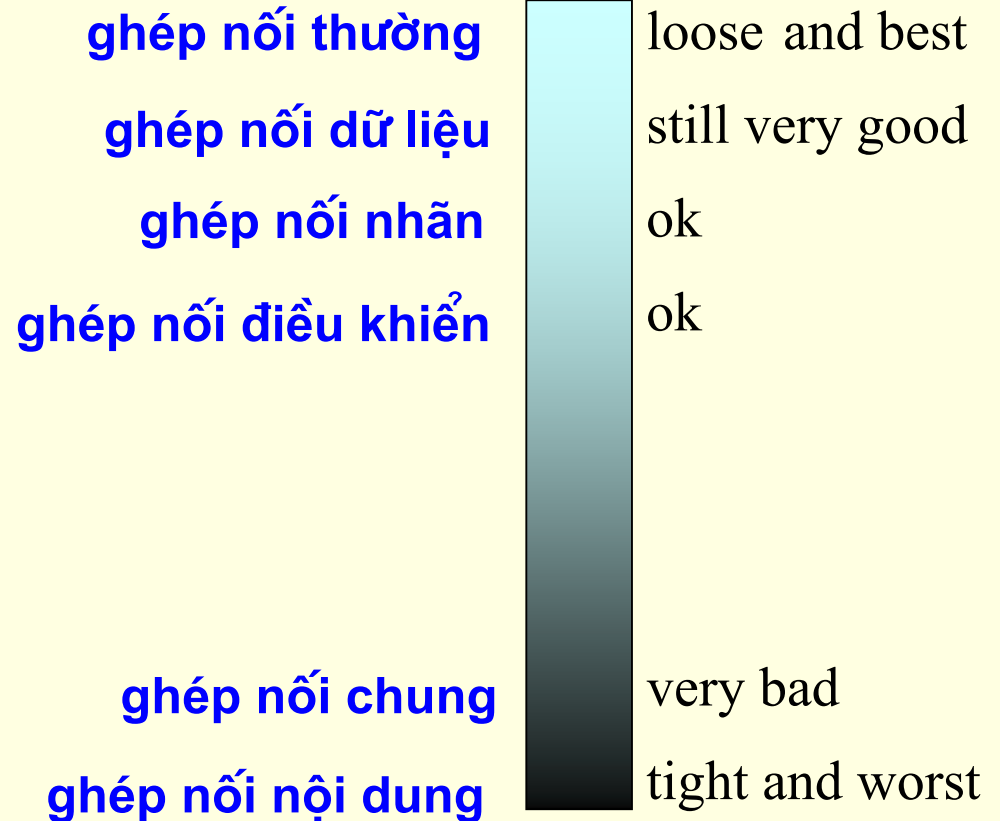
- Coupling (ghép nối)
  - độ đo sự liên kết (trao đổi dữ liệu) giữa các mô đun
  - ghép nối chặt chẽ thì khó hiểu, khó sửa đổi do phải tính đến các liên kết có thể, dễ gây lỗi lan truyền.
- Cohesion (kết dính)
  - độ đo sự phụ thuộc lẫn nhau của các thành phần trong một module
  - kết dính cao thì tính cục bộ cao (độc lập chức năng); dễ hiểu, dễ sửa đổi.
- Tiêu chuẩn của thiết kế tốt: **kết dính chặt, ghép nối lỏng**

# Ghép nối - *Coupling*



mức độ quan hệ của các module:

- module nên ghép nối lỏng lẻo
- Mức lỏng lẻo thể hiện qua loại hình ghép nối (hình bên)



# Ghép nối nội dung

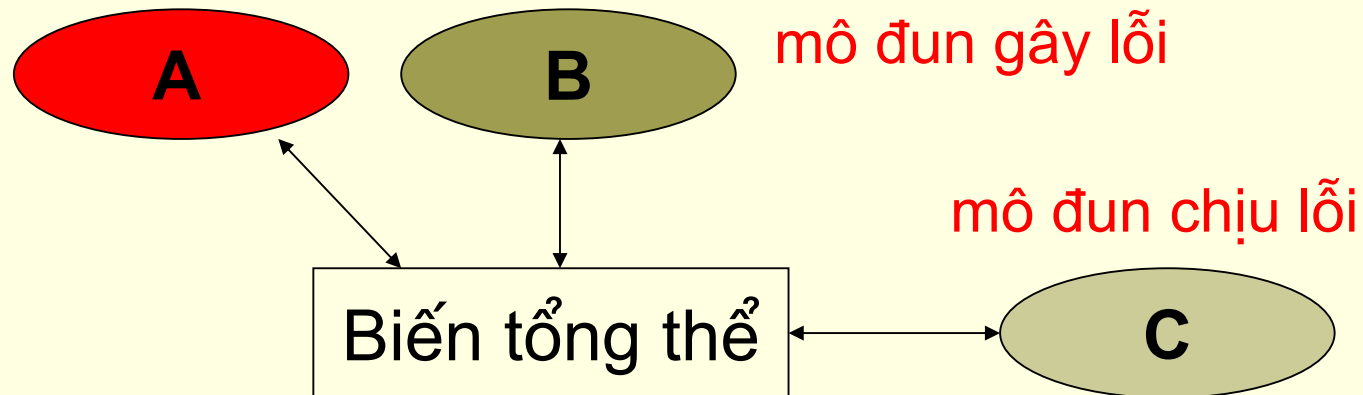
- Các module dùng dữ liệu hay thông tin điều khiển được duy trì trong 1 mô đun khác
- Là trường hợp xấu nhất. Ví dụ:
  - các ngôn ngữ bậc thấp chỉ dùng biến chung
  - lạm dụng lệnh Goto trong một chu trình

```
10 k =1
20 gosub 100
30 if y > 120 goto 60
40 k = k+1
50 goto 20
```

```
60 print k, y
70 stop
100 Y =3*k*k+7*k-3
110 return
```

# Ghép nối chung

- ◆ 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





# Ghép nối điều khiển

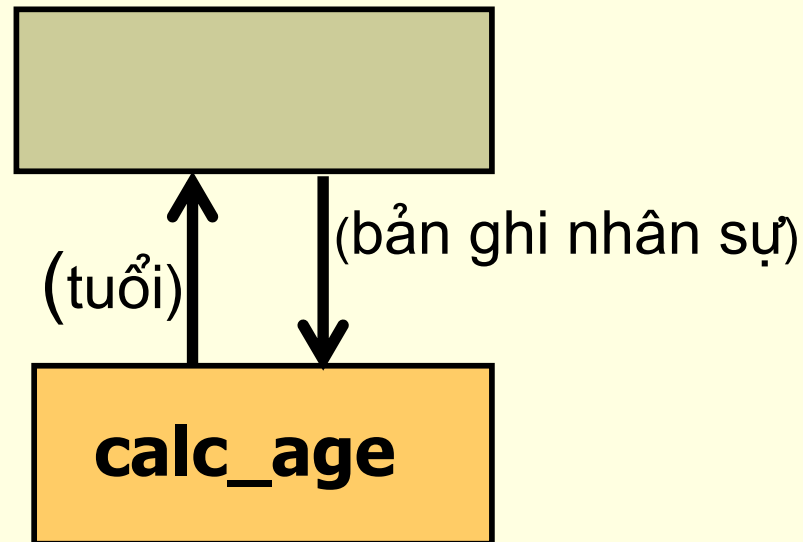
- Các module trao đổi thông tin điều khiển
- Làm cho thiết kế khó hiểu, khó sửa đổi, dễ nhầm

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

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

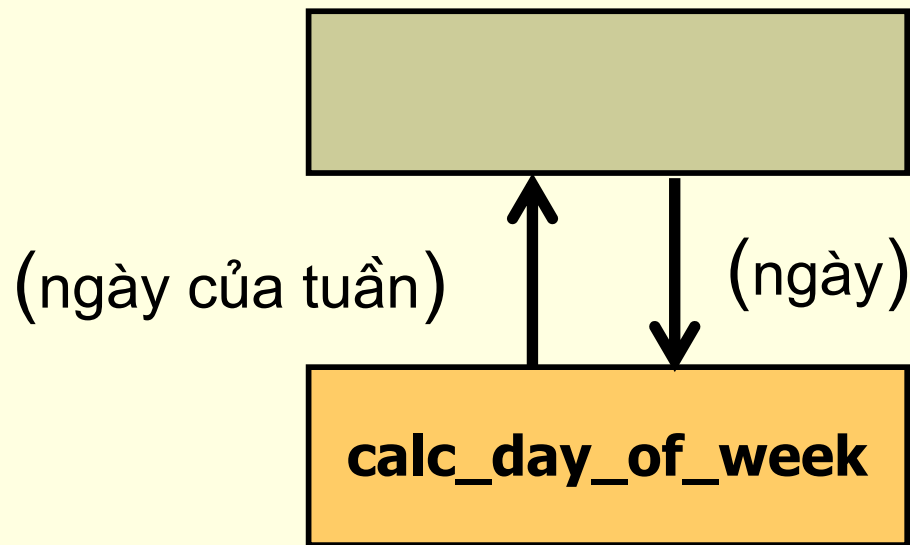
# Ghép nối nhẵn

- Các môđun trao đổi thừa thông tin
- Môđun có thể thực hiện chức năng ngoài ý muốn
- Làm giảm tính thích nghi



# Ghép nối dữ liệu

- Truyền dữ liệu qua tham số
- Nhận kết quả qua tham số và giá trị trả lại



# Kết dính - Cohesion



- mỗi môđun chỉ nên thực hiện 1 chức năng
- mọi thành phần của môđun phải tham gia thực hiện chức năng đó

chức năng  
tuần tự  
truyền thông  
thủ tục  
thời điểm  
logic

gom góp

high and best

ok

still ok

not bad at all

still not bad at all

still not bad at all

lowest and worst by far

# Các loại kết dính



- ☐ Kết dính gom góp (coincidental cohesion):
  - ◆ gom các thành phần không liên quan đến nhau
- ☐ Kết dính lôgic (logical cohesion)
  - ◆ gồm 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...)
- ☐ Kết dính thủ tục (procedural cohesion)
  - ◆ các thành phần thực hiện theo 1 thứ tự xác định (vd: tính lương cơ bản, tính phụ cấp, tính bảo hiểm)

# Các loại kết dính(t)



- Kết dính truyền thông (communicational cohesion)
  - ◆ các thành phần truy cập đến cùng tập dữ liệu: tính toán 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: ảnh màu -> đen trắng -> ảnh nén
- Kết dính chức năng (functional cohesion)
  - ◆ các thành phần cùng góp phần thực hiện một chức năng (vd: các thao tác sắp xếp)

# Tính hiểu được - *Understandability*



- Là kết quả tổng hợp từ nhiều thuộc tính
  - ◆ Cấu trúc rõ ràng, tốt
  - ◆ Ghép nối lỏng lẻo
  - ◆ Kết dính cao
  - ◆ Được lập tài liệu
  - ◆ Thuật toán, cấu trúc dễ hiểu

# Tính thích nghi được (*Adaptability*)



- Hiểu được
  - sửa đổi được, tái sử dụng được
- Tự chứa
  - không sử dụng thư viện ngoài
  - mâu thuẫn với xu hướng tái sử dụng



# Thiết kế hướng đối tượng và chất lượng



- Thiết kế hướng đối tượng hướng tới chất lượng thiết kế tốt
  - ◆ đóng gói, che dấu thông tin (độc lập dữ liệu)
  - ◆ các thực thể hoạt động độc lập (cục bộ, dùng lại)
  - ◆ trao đổi dữ liệu qua truyền thông (liên kết yếu)
  - ◆ có khả năng kế thừa (dùng lại)
  - ◆ cục bộ, dễ hiểu, dễ tái sử dụng

# Câu hỏi ôn tập



1. Thiết kế phần mềm là gì?
2. Nêu các nguyên lý thiết kế phần mềm?
3. Nêu các loại thiết kế và giải thích nội dung của nó?
4. Giải thích một số khái niệm cơ bản của thiết kế:
  - trừu tượng?
  - làm mịn?
  - mô đun hoá?
  - thủ tục?
  - che dấu thông tin?
5. Vẽ sơ đồ mô tả mối quan hệ giữa số mô đun và chi phí phát triển?

# Câu hỏi ôn tập



6. Các đặc trưng của một thiết kế tốt?
7. Các tiêu chí kỹ thuật đánh giá một thiết kế tốt
8. Lợi ích của hệ thống có kiến trúc tốt
9. Lợi ích của việc mô đun hoá trong thiết kế phần mềm là gì?
10. Lợi ích của việc che dấu thông tin là gì?
11. Có các độ đo chất lượng thiết kế nào?

# Câu hỏi ôn tập



12. Ghép nối là gì? Kể các loại ghép nối theo mức độ chặt (tôi) dần?
13. Kết dính là gì? Kể các loại kết dính theo mức độ chặt giảm (kém) dần?
14. Thế nào là tính hiểu được?
15. Thế nào là tính thích nghi được?
16. Thiết kế hướng đối tượng hướng đến chất lượng tốt ở những mặt nào?

# Câu hỏi và thảo luận

