Nhập môn **Công nghệ Phần mềm**
(Introduction to **Software Engineering**)

PGS.TS. Phan Huy Khánh
khanhph29@gmail.com, phkhanh@dut.udu.vn

Chương 3. Thiết kế hệ thống

Chương 3. Thiết kế hệ thống

- Khái niệm Thiết kế Hệ thống (TKHT)
- Vai trò, nguyên lý và chất lượng TKHT
- Quy trình thiết kế hệ thống
  - Phương pháp thiết kế cấu trúc
  - Phương pháp thiết kế hướng đối tượng
- Thiết kế kiến trúc phần mềm
- Thiết kế giao diện phần mềm

Khái niệm thiết kế hệ thống

- Thiết kế hệ thống (TKHT) :
  - Là thiết kế cấu hình phần cứng và cấu trúc phần mềm để có được hệ thống thỏa mãn các yêu cầu đề ra
  - Là thiết kế cấu trúc (What), không phải là thiết kế Logic (How)
- Cấu trúc phần mềm bao gồm :
  - Chức năng xử lý
  - Tổ chức dữ liệu

Vai trò thiết kế hệ thống

- Vai trò của TKHT :
  - Cung cấp cái nhìn từ tổng thể đến mức chi tiết của HTPM
  - Là phương tiện để trao đổi thông tin, đảm bảo chất lượng
  - Giúp người phát triển dễ hiểu, dễ kiểm chứng, bảo trì
- TKHT tạo ra tập các đơn thể chương trình (mô-đun) theo một cấu trúc phân cấp, tương tác lẫn nhau
- Để nâng cao chất lượng, khi TKHT cần xác định được :
  - Chức năng và mô hình dữ liệu của các đơn thể
  - Cách thức cài đặt đơn thể
  - Tương tác giữa các đơn thể

Nguyên lý của TKHT

- TKHT phải đảm bảo được :
  - Có cái nhìn khái quát về hệ thống, không bị bó buộc vào cái nhìn hạn hẹp
  - Có sự lựa chọn từ nhiều giải pháp khác nhau
  - Có cấu trúc đơn thể, phân cấp dễ dàng sửa đổi
  - Có khả năng quay lui lại bước phân tích yêu cầu
  - Kiểm tra được sự thỏa mãn của các yêu cầu
- Biểu diễn kết quả TKHT có tính nhất quán và tính tích hợp
  - Các đơn thể và các yêu cầu không có tương ứng một-một
  - Thiết kế do nhiều người tiến hành song song
  - Thống nhất quan điểm sử dụng, thống nhất giao diện tương tác

Cấu trúc tổng thể của hệ thống

- Cấu trúc tổng thể của hệ thống thực chất là mô tả các đơn thể và hoạt động của chúng
- Mỗi đơn thể :
  - Là dãy các lệnh thực hiện một chức năng (Function) nào đó
  - Có thể được biên dịch độc lập
  - Các đơn thể có thể có thể gọi lẫn nhau qua các giao diện
  - Giao diện là danh sách các tham biến (Arguments)
- Các công cụ thiết kế thủ tục thường gặp :
  - Mã giả, hay giả ngữ (Pseudo Code)
  - Sơ đồ (biểu đồ) luồng (Flow Chart)
  - Biểu đồ (diagram) Nassi-Shneiderman
  - Ngôn ngữ Java trong thiết kế web (JSP-Java Server Pages)

Giao diện giữa các đơn thể

- Giao diện là cách liên kết các đơn thể :
  - Quản lý tham biến hình thức/thực sự và kết quả trả về
  - Trao đổi thông tin, hạn chế dùng chung dữ liệu
  - Sử dụng kỹ thuật che giấu thông tin
- Ưu điểm khi sử dụng kỹ thuật che giấu thông tin :
  - Không cần biết cách thức cài đặt thực tế một đơn thể
  - Chỉ quan tâm đến chức năng và trao đổi dữ liệu
  - Cho phép quản lý các đơn thể độc lập, các thiết bị vào-ra…
  - Giảm hiệu ứng phụ mỗi khi sửa đổi đơn thể
  - Quản lý được tài nguyên hệ thống

Tổ chức, cấu trúc dữ liệu

- Cấu trúc dữ liệu của một hệ thống gồm :
  - Các thực thể, hay đối tượng dữ liệu
  - Mối quan hệ giữa các thực thể
  - Cách biểu diễn các thành phần
- Các mức thiết kế cấu trúc dữ liệu :
  - Thiết kế mức ý niệm, lôgic
    - Đặc tả các thực thể/đối tượng dữ liệu và quan hệ/kết hợp
    - Đặc tả các đặc tính/thuộc tính, các khóa
    - Các ràng buộc và tính chất của quan hệ
  - Thiết kế mức vật lý
    - Mô tả cụ thể (vật lý) các tệp CSDL
    - Các kiểu dữ liệu
    - Các miền giá trị và các ràng buộc sử dụng dữ liệu

Một số khái niệm thiết kế cơ sở

- Trừu tượng hóa (Abstraction) :
  - Trừu tượng hóa dữ liệu (thuộc tính)
  - Trừu tượng hóa thủ tục (phương thức)
  - Trừu tượng hóa điều khiển
- Làm mịn :
  - Biến đổi mô hình thiết kế trừu tượng thành đơn thể vật lý
- Thiết kế đơn thể :
  - Cụ thể hóa chức năng xử lý và vận hành dữ liệu mỗi đơn thể
- Kiến trúc :
  - Cấu trúc tổng thể của HTPM
- Thủ tục/hàm :
  - Thuật toán thực hiện một chức năng xử lý
- Che dấu thông tin, kích hoạt hệ thống qua giao diện

Trừu tượng hóa (Abstraction)

- Trừu tượng hóa :
  - Khái niệm cơ sở trong tư duy của con người
  - Là quá trình ánh xạ một sự vật, hay hiện tượng của thế giới thực thành một khái niệm logic
  - Tùy theo nhu cầu mà có nhiều mức khác nhau
- Vai trò của trừu tượng hóa :
  - Cho phép tập trung (tư duy) để giải quyết vấn đề mà không bận tâm đến chi tiết
  - Biểu diễn vấn đề bằng một cấu trúc tự nhiên

Ví dụ trừu tượng dữ liệu và thủ tục

Trừu tượng dữ liệu

Trừu tượng thủ tục

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*

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 cửa 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

Chất lượng TKHT

Một TKHT tốt thường thể hiện được ba đặc trưng sau :

- Tính rõ ràng, tường minh :
  - Triển khai được mọi yêu cầu mà khách hàng đòi hỏi
  - Tuân thủ các đặc tả trong kết quả phân tích hệ thống
- Tính dễ đọc, dễ hiểu :
  - Thiết kế có hướng dẫn, dễ đọc, dễ hiểu cho người lập trình, người kiểm thử và người bảo trì
- Tính đầy đủ, toàn vẹn :
  - Thiết kế cung cấp một bức tranh đầy đủ về HTPM hướng đến các yếu tố dữ liệu, chức năng xử lý và hành vi của hệ thống

Độ đo chất lượng TKHT

- Độ đo chất lượng TKHT phụ thuộc vào bài toán cụ thể, không có phương pháp chung
- Độ đo chất lượng thường áp dụng cho các đơn thể
- Một số độ đo chất lượng thường gặp :
  - Mức độ ghép nối giữa các đơn thể (Coupling)
  - Mức độ kết dính giữa các thành phần bên trong một đơn thể (Cohesion)
  - Tính hiểu được (Understandability)
  - Tính thích nghi được (Adaptability)

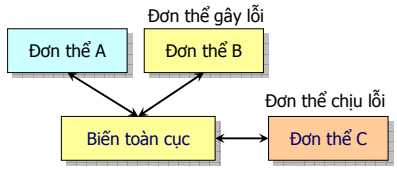
Vận dụng độ đo trong TKHT

- Độ đo ghép nối (Coupling)
  - Đo khả năng liên kết (trao đổi dữ liệu) giữa các đơn thể
  - Ghép nối chặt thường gây khó hiểu, khó sửa đổi do phải tính hết mọi liên kết có thể, dễ gây lỗi lan truyền
  - Thực tế nên ghép nối lỏng lẻo (Loose) các đơn thể
- Độ đo kết dính (Cohesion)
  - Đo sự phụ thuộc lẫn nhau của các thành phần trong cùng một đơn thể
  - Kết dính chặt (cao) thì tính cục bộ cao (độc lập chức năng), thường dễ hiểu, dễ sửa đổi
- Tiêu chuẩn của thiết kế tốt : kết dính chặt, ghép nối lỏng

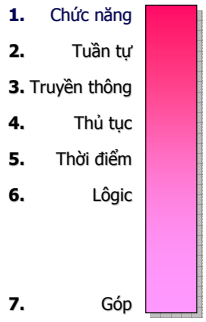
Mức độ ghép nối các đơn thể

Ghép nối chung

- Trong ghép nối chung các đơn thể :
  - Dữ liệu trao đổi thông qua biến toàn cục (Global Variables)
  - Lỗi của một đơn thể này có thể ảnh hưởng đến hoạt động của đơn thể khác
  - Thường khó sử dụng lại các đơn thể



Bảy mức độ kết dính các đơn thể



Mọi thành phần của mỗi đơn thể chỉ nên thực hiện một chức năng

Giải thích các loại kết dính

1. **Kết dính góp** (Coincidental Cohesion):
   Gom các thành phần không liên quan đến nhau
2. **Kết dính lôgic** (Logical Cohesion)
   Gồm các thành phần có cùng chức năng lôgic
3. **Kết dính thời điểm** (Temporal Cohesion)
   Gồm các thành phần hoạt động cùng thời điểm
4. **Kết dính thủ tục** (Procedural Cohesion)
   Gồm các thành phần thực hiện theo một thứ tự xác định
5. **Kết dính truyền thông** (Communicational Cohesion)
   Gồm các thành phần truy cập đến cùng tập dữ liệu
6. **Kết dính tuần tự** (Sequential Cohesion)
   Cái ra của một thành phần là cái vào của thành phần tiếp theo
7. **Kết dính chức năng** (Functional Cohesion)
   Gồm các thành phần cùng thực hiện một chức năng

Tính hiểu được

- Tính hiểu được (Understandability)
  thể hiện qua các tiêu chí đạt được của hệ thống
- Bao gồm các tiêu chí :
  - Cấu trúc rõ ràng, chặt chẽ
  - Thuật toán xử lý dễ hiểu, dễ triển khai
  - Cấu trúc dữ liệu hợp lý
  - Ghép nối lỏng lẻo
  - Tính kết dính cao
  - Có tài liệu giải thích, hướng dẫn chi tiết

Tính thích nghi được

- Tính thích nghi được (Adaptability)
  thể hiện qua các tiêu chí đạt được của hệ thống
- Bao gồm :
  - Tính tương thích :
    - Khả năng sửa đổi được
    - Tái sử dụng được
  - Tính tự chứa :
    - Không sử dụng thư viện bên ngoài
    - Không xảy nra mâu thuẫn với xu hướng tái sử dụng

Quy trình thiết kế hệ thống

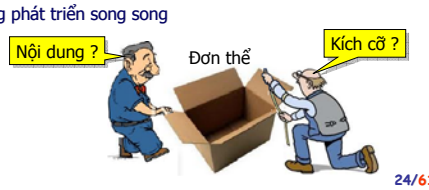
- Quy trình thiết kế hệ thống bao gồm các bước :
  - Phân chia mô hình phân tích thành các hệ thống con
    theo nguyên lý chia để trị
  - Tìm sự tương tranh (Concurrency) có thể xảy ra
    giữa các hệ thống con trong hệ thống
  - Thiết kế giao diện, hay giao tiếp với NSD
  - Chọn chiến lược cài đặt, quản trị cơ sở dữ liệu (CSDL)
  - Tìm nguồn tài nguyên chung và cơ chế điều khiển truy cập
  - Xây dựng kịch bản sử dụng hệ thống
- Phương pháp TKHT thường được sử dụng hiện nay :
  - Thiết kế cấu trúc (Structured Design)
  - Thiết kế hướng đối tượng (Object-Oriented Design)

Phương pháp thiết kế cấu trúc

- Thiết kế cấu trúc mô tả :
  - Cấu trúc tổng thể của hệ thống gồm các đơn thể, thành phần
  - Giao diện, hay mối quan hệ-tương tác, giữa các đơn thể
  - Tổ chức, cấu trúc dữ liệu
- Không cần phải chỉ ra trong thiết kế cấu trúc :
  - Thứ tự thực hiện các đơn thể
  - Số lần thực hiện cho mỗi đơn thể
  - Chi tiết thiết kế của từng đơn thể
- Trong thiết kế cấu trúc, người ta thường sử dụng
  biểu đồ cấu trúc (Structure Chart)

Tổ chức đơn thể

- Khái niệm đơn thể dựa trên quan điểm chia để trị :
  - Mỗi đơn thể thực hiện một công việc nào đó
  - Kích cỡ đơn thể nên được quyết định
    dựa trên khái niệm độc lập chức năng :
- Ưu điểm :
  - Giảm độ phức tạp
  - Cục bộ, dễ sửa đổi nên dễ tái sử dụng
  - Có khả năng phát triển song song



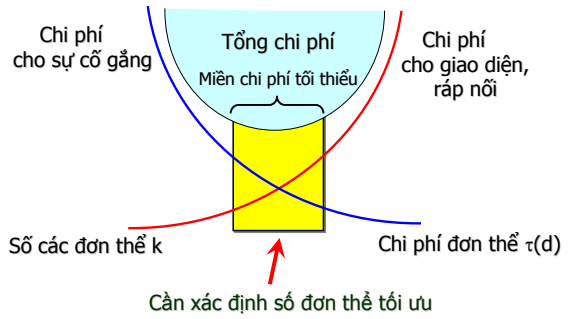
Chi phí của PP chia để trị

- Độ phức tạp PP chia để trị được tính toán như sau :
  - Giả sử bài toán **P** sử dụng **d** dữ liệu
  - Gọi **τ(d)** là thời gian thực hiện phép tính trên **d** dữ liệu
  - Bài toán được chia ra thành **k** bài toán nhỏ hơn giống nhau
- Giả sử d chia hết cho k, khi đó :
  - **τ(d) = k * τ(d/k) + C(k, d)**
  - Với **C(k, d)** là thời gian cần thiết để hoàn thành ráp nối tất cả **k** bài toán nhỏ hơn giống nhau
- Rõ ràng, **k** càng lớn thì thời gian cần thiết cho **C(k, d)** cũng tăng theo

Lập luận về độ phức tạp chia để trị

- Giả sử cho bài toán P sử dụng PP chia để trị
- Cho hai hàm :
  - C(P) là độ phức tạp cảm nhận được của P
  - E(P) là độ cố gắng theo thời gian để giải P
- Khi đó, nếu P_1 và P_2 là hai bài toán (đơn thể) thì $C(P_1) > C(P_2) \rightarrow E(P_1) > E(P_2)$
- Thực tế :
  - $C(P_1 + P_2) > C(P_1) + C(P_2)$
- Từ đó :
  - $E(P_1 + P_2) > E(P_1) + E(P_2)$

Mô hình tính độ phức tạp chia để trị



Thiết kế hướng đối tượng (HĐT)

- Thiết kế HĐT :
  - Hiện nay trở nên phổ biến, là giải pháp cho các hệ thống lớn, phức tạp
  - là một cách tiếp cận khác với thiết kế hướng thủ tục
- Thiết kế HĐT nhìn nhận hệ thống theo quan điểm :
  - Hệ thống là tập các đối tượng tương tác với nhau
  - Mỗi đối tượng đóng gói hai thành phần :
    - Thuộc tính (dữ liệu) và phương thức (xử lý dữ liệu)
  - Tương tác giữa các đối tượng bằng cách truyền thông báo hay thông điệp (Messages)
  - Các đối tượng có thể kế thừa nhau

So sánh với thiết kế hướng thủ tục

- Trong thiết kế hướng thủ tục :
  - Cấu trúc dữ liệu/CSDL dùng chung cho cả hệ thống
  - Mọi thủ tục thao tác trên CSDL chung có chung trạng thái
  - Một thủ tục gây lỗi trên dữ liệu đang xử lý, hoặc sửa đổi một thủ tục có thể lan truyền, ảnh hưởng sang các thành phần khác của hệ thống
- Hạn chế :
  - Thay đổi cấu trúc dữ liệu dẫn đến thay đổi tổng thể hệ thống, do đó cần tổ chức tốt dữ liệu
  - Với các hệ thống càng lớn, càng phức tạp, việc bảo trì càng khó khăn, chi phí càng cao

Ưu điểm của thiết kế HĐT

- Thiết kế HĐT phù hợp cho các hệ thống lớn, phức tạp
- Các ưu đểm nổi bật :
  - Đảm bảo tính độc lập dữ liệu do sử dụng các nguyên lý đóng gói, che dấu thông tin :
  - Các đối tượng là các thực thể hoạt động độc lập, cục bộ
  - Trao đổi dữ liệu qua truyền thông, nguyên lý đa
  - Có khả năng kế thừa, dùng lại được
  - Dễ hiểu
  - Dễ bảo trì

Khái niệm kiến trúc phần mềm

- Kiến trúc phần mềm (KTPM) :
  - Mô tả cấu trúc tổng thể của một phần mềm
  - Thể hiện dạng một biểu đồ phân cấp gồm các thành phần và mối quan hệ giữa chúng
  - Cho phép nhìn hệ thống theo nhiều góc nhìn khác nhau :
    - Góc nhìn tĩnh
    - Góc nhìn động
    - Góc nhìn dữ liệu
    - Góc nhìn triển khai
- Kiến trúc phần mềm được đặc tả ngay từ giai đoạn đầu của thiết kế hệ thống

Các bước thiết kế kiến trúc phần mềm

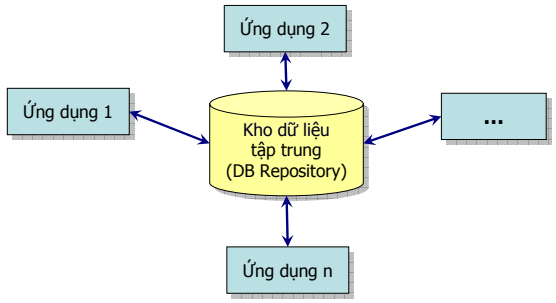
- Gồm các bước :
  - Cấu trúc hóa hệ thống :
    - Phân chia hệ thống thành các hệ con (Sub-System) độc lập
    - Xác định giao diện trao đổi thông tin giữa chúng
  - Mô hình hóa điều khiển :
    - Xác lập mô hình điều khiển giữa các thành phần khác nhau của hệ thống
  - Phân rã thành các đơn thể :
    - Phân rã các hệ con thành các đơn thể
- Quá trình thiết kế tạo ra các mô hình kiến trúc khác nhau

Các mô hình kiến trúc

- Mỗi mô hình kiến trúc biểu diễn một cách nhìn hệ thống
- Gồm các mô hình :
  - Kiến trúc dữ liệu tập trung (Data-Centered Architectures)
  - Kiến trúc khách / dịch vụ (Client-Server Architectures)
  - Kiến trúc phân tầng (Layered Architectures)
  - Kiến trúc gọi và trả lại (Call and Return Architectures)
  - Kiến trúc luồng dữ liệu (Data Flow Architectures)
  - Kiến trúc hướng đối tượng (Object-Oriented Architectures)

Kiến trúc dữ liệu tập trung

- Các ứng dụng (phần mềm khách) khác nhau cùng dùng chung, cùng chia sẻ một kho dữ liệu tập trung

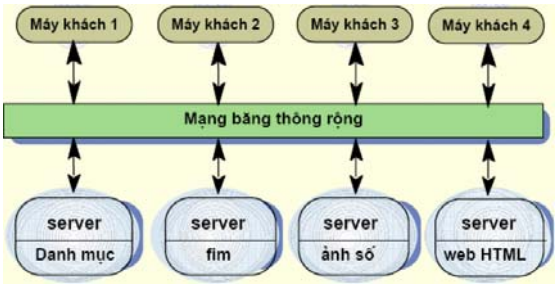


Đánh giá kiến trúc dữ liệu tập trung

- Ưu điểm
  - Tiện lợi cho chia sẻ dữ liệu lớn
  - Các ứng dụng sử dụng kho DL không cần biết dữ liệu được tạo ra, cập nhật và quản lý như thế nào
- Nhược điểm
  - Các hệ con phải tuân theo mô hình dữ liệu của kho
  - Việc quản lý kho DL khó khăn, phức tạp, chi phí đắt đỏ
  - Khó có chính sách quản lý DL riêng cho các hệ con
  - Khó phân bố dữ liệu một cách hiệu quả giữa các ứng dụng

Mô hình kiến trúc khách/dịch vụ

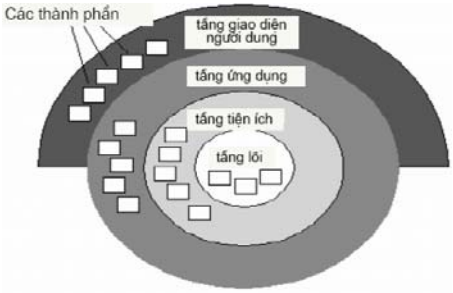
- Giữa các máy khách (chạy các ứng dụng Clients) và các máy dịch vụ (Server) là hệ thống mạng



Đánh giá kiến trúc khách/dịch vụ

- Ưu điểm
  - Phân phối dữ liệu trực tiếp
  - Sử dụng hiệu quả mạng, tận dụng được nhiều loại thiết bị
  - Dễ dàng mở rộng, nâng cấp, thêm mới dịch vụ
- Nhược điểm
  - Các hệ con dùng cấu trúc dữ liệu khác nhau, không chia sẻ được với nhau, trao đổi dữ liệu có thể không hiệu quả
  - Quản lý ở các máy dịch vụ nhiều khi trùng lặp, dư thừa
  - Khó quản lý ứng dụng, khó tìm máy dịch vụ rỗi để giảm chi phí chờ đợi phục vụ

Mô hình kiến trúc phân tầng



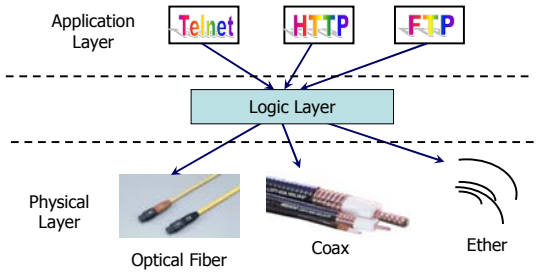
Abstract machine model

Kiến trúc phân tầng

- Bản chất kiến trúc phân tầng :
  - Phân rã hệ thống thành các tầng, mỗi tầng là một tập hợp các dịch vụ khác nhau
  - Dùng để mô hình hóa giao diện của các phân hệ (Sub-Systems)
  - Trợ giúp phát triển các tầng, khi giao diện mỗi tầng thay đổi thì chỉ ảnh hưởng tới các tầng liền kề
- Đặc điểm kiến trúc phân tầng :
  - Giúp bảo trì, cập nhật hệ thống dễ dàng
  - Không phải hệ thống nào cũng dễ dàng xây dựng được kiến trúc phân tầng

Lợi ích của kiến trúc phân tầng

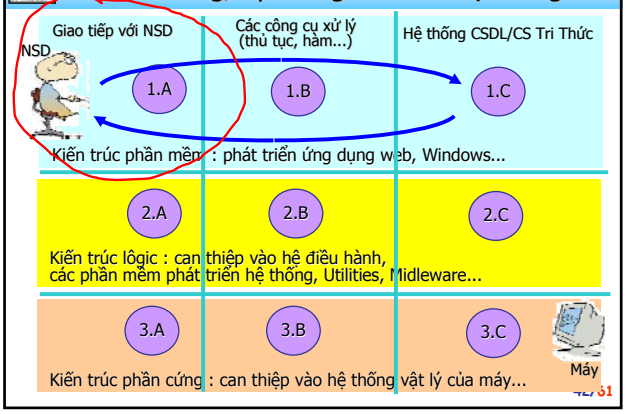
- Trong các ứng dụng mạng :
  - Các giao thức được phân theo tầng, để giảm bớt trao đổi thông tin giữa các thực thể trong một hệ thống



Kiến trúc phân tầng OSI

- Mô hình OSI hay mô hình tham chiếu kết nối các hệ thống mở
- Tiếng Anh : Open Systems Interconnection Reference Model, (viết gọn OSI Model hoặc OSI Reference Model)

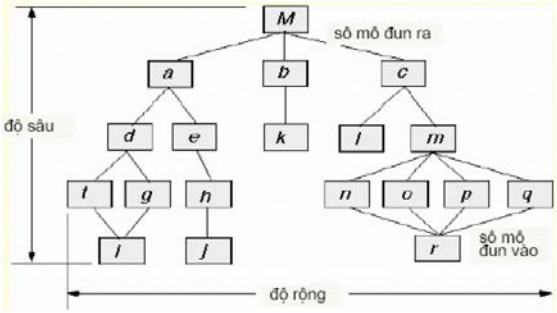
Phân tầng/lớp trong thiết kế hệ thống



Kiến trúc gọi và trả lại

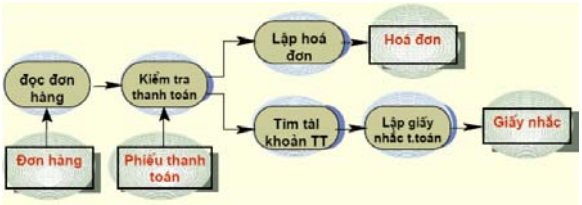
- Trong kiến trúc gọi - trả lại :
  - Bao gồm chương trình chính (Main Program) và các chương trình con (Sub-Program)
  - Các hệ thống con chia thành những đơn thể (Modules)
  - Một đơn thể là một thành phần của hệ thống, phối hợp với những thành phần khác để cung cấp dịch vụ
  - Mỗi hệ thống con có vai trò hoạt động độc lập, không phụ thuộc vào dịch vụ do các hệ thống con khác cung cấp

Mô hình kiến trúc gọi và trả lại



Kiến trúc luồng dữ liệu

- Ví dụ hệ thống xử lý đơn mua hàng



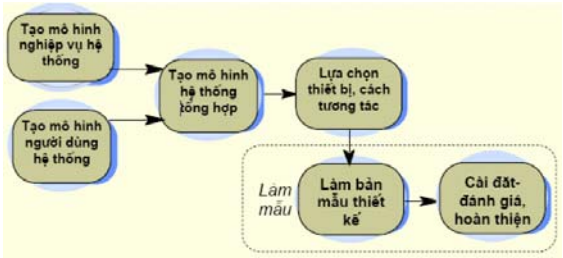
Thiết kế giao diện

- Thiết kế giao diện gồm các nội dung :
  - Vai trò, tầm quan trọng
  - Tiến trình thiết kế giao diện chung
  - Tiến trình thiết kế giao diện làm mẫu
  - Nguyên tắc thiết kế giao diện
  - Thiết bị và kiểu tương tác
  - Các loại giao diện truyền thống
  - Các hình thức tương tác
  - Một số vấn đề thiết kế

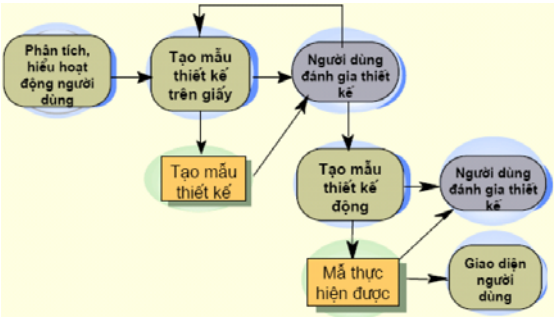
Vai trò, tầm quan trọng

- Thiết kế giao diện :
  - Một khâu không thể thiếu trong thiết kế hệ thống, NSD thường đánh giá phần mềm qua giao diện
  - Che dấu chi tiết kỹ thuật bên trong hệ thống
  - Kết hợp 3 mặt : công thái học, chức năng, công nghệ
- NSD làm việc với hệ thống thông qua giao diện
- Giao diện trợ giúp NSD làm việc với khả năng của họ
  - Giao diện trợ giúp tốt, NSD dễ thành công
  - Giao diện thiết kế nghèo nàn làm NSD khó khăn, dễ mắc lỗi
- Giao diện thiết kế tồi là một trong nhiều lý do dẫn đến phần mềm không được sử dụng

Tiến trình thiết kế giao diện chung



Tiến trình thiết kế giao diện làm mẫu



Nguyên tắc thiết kế giao diện

- Cần phản ảnh vào giao diện khi thiết kế :
  - Kinh nghiệm, năng lực, nhu cầu của NSD
  - Khả năng dùng bàn phím, chuột,...
  - Sở thích, văn hóa, lứa tuổi : mầu sắc, ngôn ngữ...
  - Tính thẩm mỹ, khoa học
  - Tính công thái học, đắc nhân tâm
- Những điểm chú ý :
  - Giải quyết tốt hạn chế về mặt vật chất, tinh thần của NSD (trí nhớ kém, thao tác vụng về, dễ mắc lỗi...)
  - Tốc độ phản ứng, khả năng nhớ thao tác của NSD ...
- Luôn bao gồm việc làm bản mẫu để NSD đánh giá

Một số yêu cầu khác

- Giao diện cần đảm bảo cho NSD :
  - Tính thân thiện :
    về thuật ngữ, khái niệm, thói quen, trình tự nghiệp vụ
  - Tính nhất quán :
    về vị trí hiển thị, câu lệnh, lệnh đơn, biểu tượng, màu sắc...
  - Ít 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 và kiểu tương tác

- Thiết bị tương tác
  - Màn hình
  - Bàn phím
  - Mouse, bút từ, ...
  - Màn hình cảm biến
  - Mic / Speaker
  - Smart cards,...
  - Bóng xoay
- Các kiểu tương tác
  - 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

- Cho đến nay, giao diện đã có nhiều tiến bộ vượt bậc nhờ :
  - Sự phát triển nhanh chóng của công nghệ phần cứng
  - Tiến bộ của Khoa học-Công nghệ
  - Sự phát triển của tư duy con người,
    kinh nghiệm ứng dụng CNTT trong cuộc sống
- Người ta thường chia ra hai loại giao diện :
  - Giao diện truyền thống cho các ứng dụng trên các HĐH Unix, Linux, MS-DOS... trước đây
  - Giao diện đồ họa (GUI – Grapic User Interface)

Giao diện truyền thống

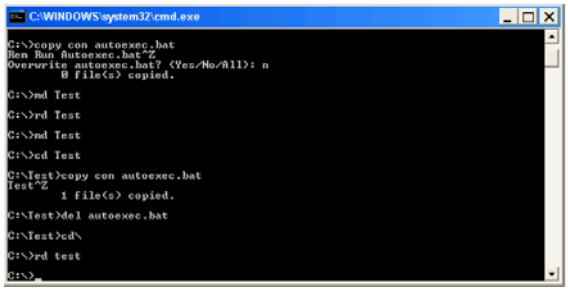
- Thường dùng cho các hệ thống tương tác đơn nhiệm, là phương thức tương tác có sớm nhất
- Cách làm việc là nhập lệnh/dữ liệu trực tiếp từ bàn phím thông qua các dòng lệnh
- Bản chất của giao diện truyền thống :
  - Thực hiện thông qua các hàm/thủ tục thư viện của ngôn ngữ
  - Có khả năng tổ hợp lệnh để tạo các lệnh phức tạp (Macro)
  - Phối hợp các Filter, tạo các lô xử lý (Batch)
  - Có thể lập trình qua Shell (chẳng hạn HĐH Unix)
  - Có thể tự động hóa
  - It tiêu tốn tài nguyên, nguồn lực của hệ thống

Đánh giá giao diện truyền thống

- Ưu điểm của giao diện truyền thống
  - Lập trình đơn giản
  - Dễ cài đặt hớn so với GUI
- Hạn chế :
  - Thao tác thực hiện tuần tự
  - Khó sửa lỗi thao tác trước đó
  - Không phù hợp với NSD ít kinh nghiệm
  - Khó học, khó nhớ
  - Dễ nhầm
  - Đòi hỏi kỹ năng sử dụng bàn phím

Ví dụ giao diện truyền thống

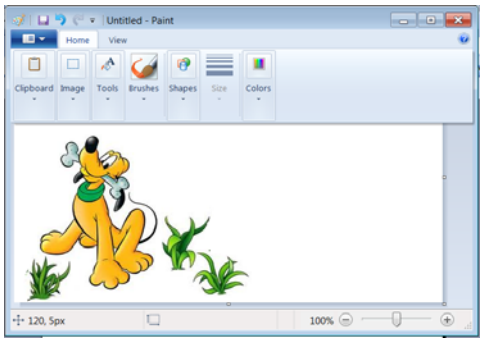
- HĐH MS-DOS và các ứng dụng trước đây



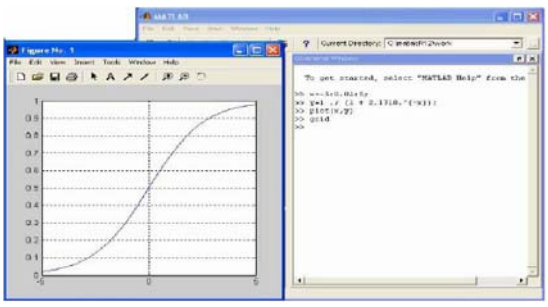
Giao diện đồ họa

- Giao diện đồ họa, GUI (Graphic User Interface)
  - Nảy sinh khi khi nâng cấp một hệ điều hành
  - Phát triển mạnh mẽ, trở nên phổ cập, quen thuộc từ khi công nghệ màn hình phát triển, có độ phân giải cao
- Liên quan đến GUI, người ta xây dựng chuẩn giao diện lập trình ứng dụng API (Application Programming Interface)
- Giao diện đồ họa có 2 hình thức tương tác chủ yếu :
  - Tương tác trực tiếp
  - Tương tác gián tiếp
- Tham khảo trang web :
  http://www.tinhte.vn/threads/673107/

Ví dụ tương tác trực tiếp



Ví dụ tương tác gián tiếp



Các hình thức giao diện

- Có ba hình thức tạo giao diện :
  - "Kéo thả" truyền thống (Pop-Up Menu) : che-hiện
    Thường gặp trong các ứng dụng Windows

File	Update	Process	View/Print	Help
Hệ thống	Cập nhật	Thống kê	InXem	Trợ giúp
Lịch công tác	Khoa	Tình hình mổ	Tháng	Giới thiệu
Máy tính	Phòng mổ	DS Bệnh nhân	Quý	Hướng dẫn
Cài đặt	Bác sĩ		Năm	
Trò chơi	Bệnh nhân			
Thoát	Sổ theo dõi mổ			

  - "Duyệt" trên trang web (Web Browser)
  - Liệt kê (hiện) tất cả (trong chừng mực có thể) theo kiểu giao diện truyền thống