HIENLTH

Implementation

Chủ đề 5: Cài đặt Phần mềm

Textbook

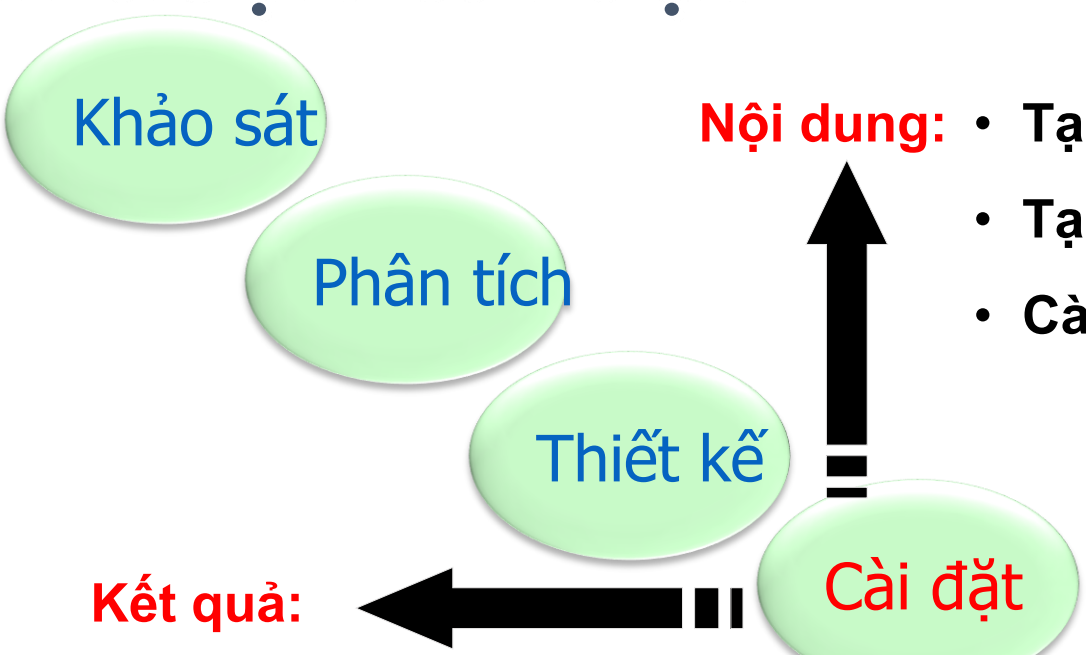
- Pressman, Software Engineering, chapter 16
- Sommerville, Software Engineering, chapter 19

Cảm ơn

Bài giảng này tham khảo từ các nguồn sau:

- Slide bài giảng CNPM, Trần Ngọc Bảo, ĐH Sư phạm TpHCM
- Slide bài giảng CNPM, Trần Anh Dũng, ĐH CNTT, ĐHQG TpHCM.

Giai đoạn cài đặt



Nội dung:

- **Tạo CSDL**
- **Tạo giao diện**
- **Cài đặt các xử lý**

Kết quả:

- **Source code:**
  - **Giao diện**
  - **DLL**
  - **ActiveX Control**
  - **Sample Database**
- **Chương trình: EXE, Web App**

Cài đặt

- Mục tiêu:
  - Biết cách sử dụng môi trường phát triển để xây dựng chương trình phần mềm
- Nội dung
  - Tổng quan
  - Phương pháp lập trình
  - Một số qui tắc lập trình
  - Mô hình 1 lớp, 2 lớp, 3 lớp

Tổng quan

- Cài đặt: Là quá trình chuyển đổi từ thiết kế chi tiết sang mã lệnh.
- Thế nào là ngôn ngữ lập trình tốt?
  - Tập trung vào nhu cầu xác định dự án phát triển của từng phần mềm riêng.
  - Có thể thiết lập được một tập hợp tổng quát những yêu cầu như sau:
    - Dễ dịch thiết kế sang chương trình
    - Có trình biên dịch hiệu quả,
    - Khả chuyển chương trình gốc
    - Có sẵn công cụ phát triển
  - Dễ bảo trì

Ngôn ngữ lập trình tốt

- Dễ dịch thiết kế sang chương trình
  - Về lý thuyết, việc sinh chương trình gốc từ một đặc tả chi tiết nên là trực tiếp.
  - Tính dễ dịch thiết kế sang chương trình sẽ cho phép một ngôn ngữ cài đặt trực tiếp cho các kết cấu có cấu trúc, các cấu trúc dữ liệu phức tạp, các vào/ra đặc biệt, khả năng thao tác bit và các đối tượng.
  - Làm cho việc dịch từ thiết kế sang chương trình gốc dễ hơn nhiều.

Ngôn ngữ lập trình tốt

- Có trình biên dịch hiệu quả
  - Mặc dầu những tiến bộ nhanh chóng trong tốc độ xử lý và mật độ nhớ đã bắt đầu làm giảm nhẹ nhu cầu chương trình siêu hiệu quả, nhiều ứng dụng vẫn còn đòi hỏi các chương trình chạy nhanh, gọn (yêu cầu bộ nhớ thấp).
  - Các ngôn ngữ với trình biên dịch tối ưu có thể là hấp dẫn nếu hiệu năng phần mềm là yêu cầu chủ chốt.

Lựa chọn NNLT

- Phụ thuộc vào cấu hình máy
- Phụ thuộc vào số lượng ngôn ngữ lập trình sẵn có
- Phụ thuộc vào thói quen sử dụng ngôn ngữ lập trình
- Phụ thuộc vào khách hàng
- …

→ Cần đánh giá rủi ro khi chọn ngôn ngữ lập trình

Kỹ năng lập trình

- Hiểu rõ ngôn ngữ (language-specific)
- Sử dụng tên biến thích hợp và có nghĩa
  - Tên biến phải rõ ràng, tránh nhầm lẫn
- Nên có các chú thích bên trong mô-đun
- Mã lệnh chuẩn
  - Thống nhất về cách đặt tên Mô-đun, tên hàm, tên biến,…
- Khả năng tái sử dụng

Kỹ năng lập trình

- Thông tin tối thiểu của một mô-đun:
  - Tên mô-đun
  - Mô tả vắn tắt các công việc mô-đun phải thực hiện
  - Tên lập trình viên
  - Ngày viết
  - Ngày chỉnh sửa
  - Danh sách các tham số
  - Danh sách các biến
  - …

Lựa chọn phương pháp lập trình?

- Lập trình tuyến tính (tuần tự)
  - Khó sửa, dễ sinh lỗi
- Lập trình có cấu trúc (thủ tục)
  - Dễ hiểu hơn, an toàn hơn
- Lập trình hướng chức năng
  - Trao đổi dữ liệu bằng tham số
  - Loại bỏ hoàn toàn dữ liệu dùng chung
- Lập trình hướng đối tượng
  - Tái sử dụng, thuận tiện cho các ứng dụng lớn
- Lập trình Logic
  - Mô tả tri thức (Prolog)

Lập trình tuyến tính

- Khi các phần mềm còn rất đơn giản:
  - Chương trình được viết tuần tự với các câu lệnh thực hiện từ đầu đến cuối.
- Tuy nhiên:
  - Khoa học máy tính ngày càng phát triển.
  - Các phần mềm đòi hỏi ngày càng phức tạp và lớn hơn rất nhiều.
- Phương pháp lập trình tuyến tính kém hiệu quả?

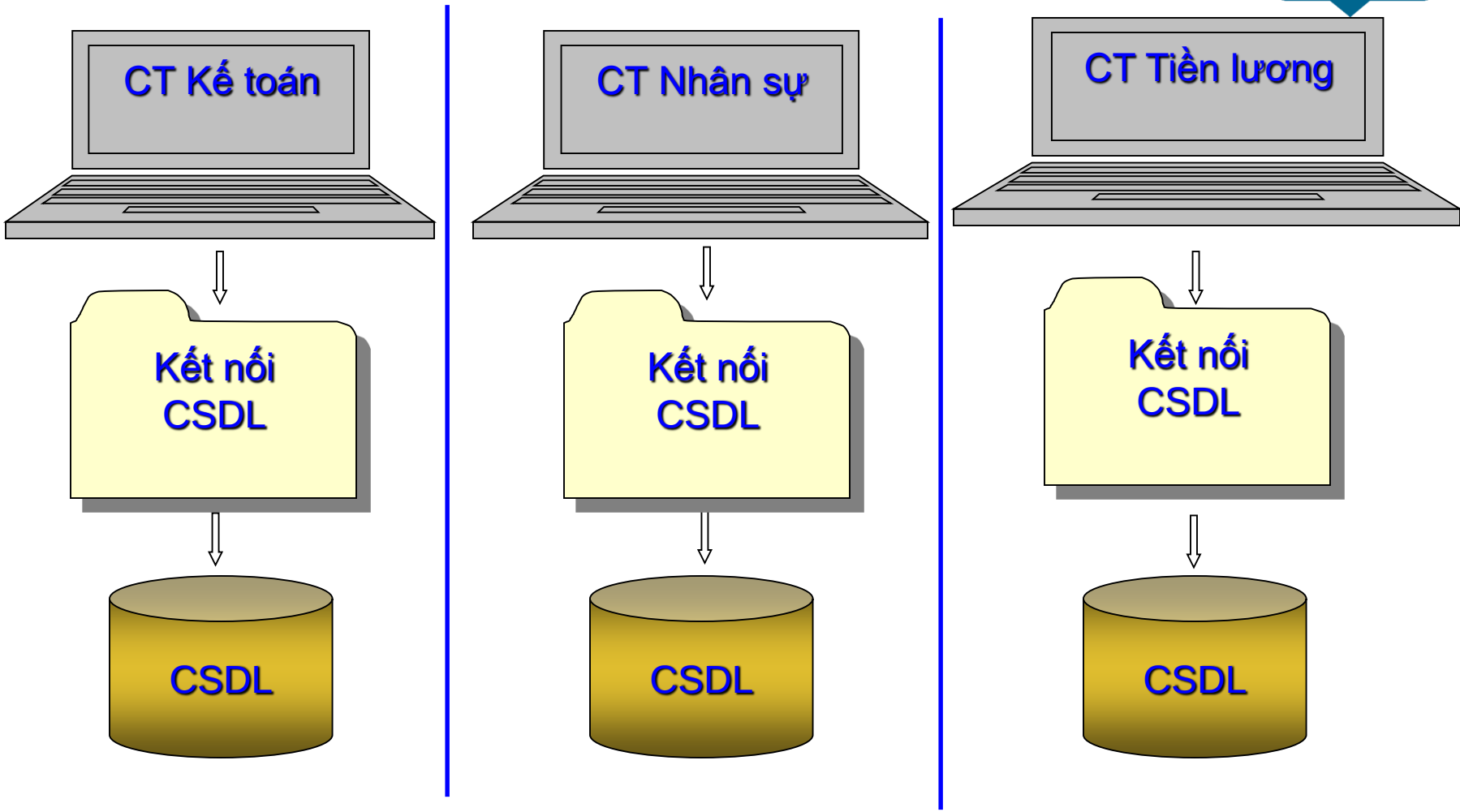
Lập trình cấu trúc

- Phương pháp lập trình thủ tục hay lập trình cấu trúc
  - Hệ thống chia các chức năng (hàm) thành các chức năng nhỏ hơn.
  - Chương trình được tổ chức thành các chương trình con
  - Chương trình = Cấu trúc dữ liệu + giải thuật
- Tổ chức dữ liệu như thế nào?
- Khi thay đổi cấu trúc dữ liệu?

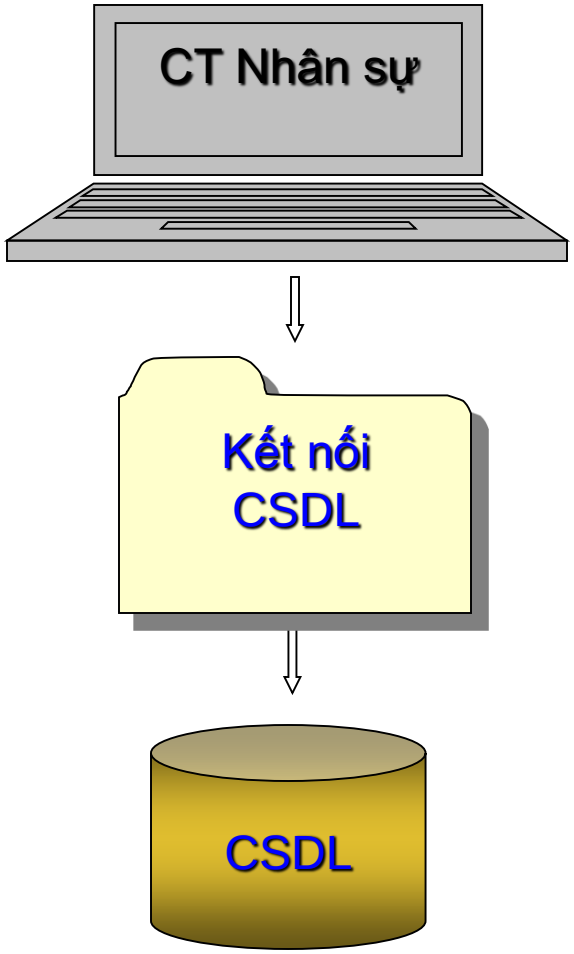
Lập trình Hướng đối tượng

- Lập trình hướng đối tượng – Lập trình định hướng đối tượng - OOP
  - Là phương pháp lập trình lấy đối tượng làm nền tảng để xây dựng thuật giải, xây dựng chương trình.
  - Dữ liệu + Hành vi của dữ liệu = Đối tượng
- Cách tiếp cận gần gũi và thực tế

CSDL trong ứng dụng quản lý

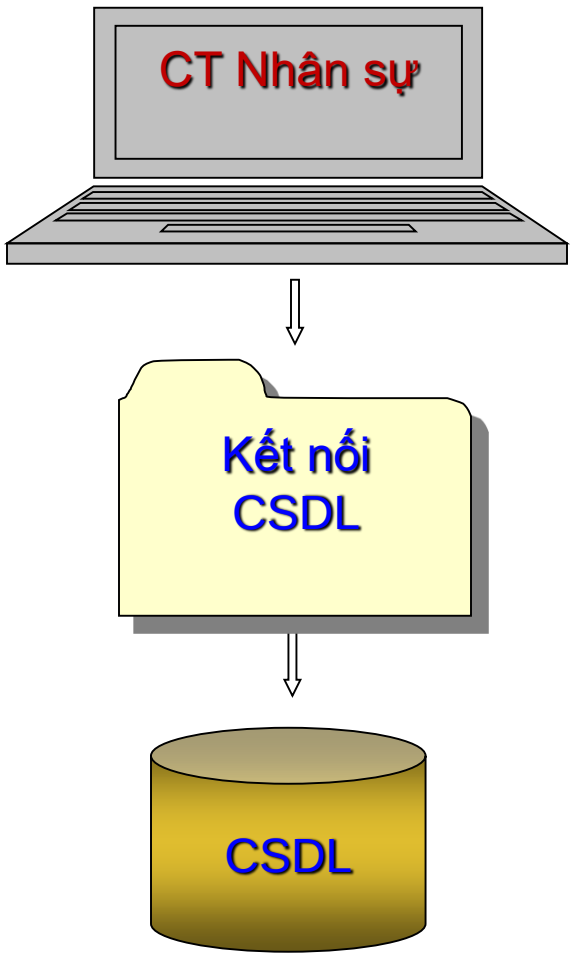


CSDL trong ứng dụng quản lý



- Thành phần giao diện
- Giao tiếp dữ liệu
- Thành phần dữ liệu

CSDL trong ứng dụng quản lý



- Thành phần giao diện: Visual Basic, Visual C++, C#, VB.NET, Delphi...
- Giao tiếp dữ liệu: ODBC, DAO, ADODB, ADO.NET,..
- Thành phần dữ liệu: MS Access, SQL Server, Oracle,...

Một số nguyên tắc trong lập trình

1. Cố gắng tái sử dụng
2. **ENFORCE INTENTIONS**

- "If your code is intended to be used in particular ways only, write it so that the code *cannot be used in any other way"*

ENFORCE INTENTIONS

- Nếu một phần tử (lớp, hàm, biến, …) không được sử dụng bởi các chức năng khác
  - Định nghĩa chúng là private hoặc protected, …
- Sử dụng các toán tử như final, abstract, …

ENFORCE INTENTIONS

- Có cái nhìn tổng thể, chương trình là cục bộ
  - Xây dựng các phần tử của chương trình:
    - cục bộ nhất có thể
    - vô hình nhất có thể
    - Các thuộc tính private
      - Truy cập chúng bằng nhiều hàm truy cập (nếu yêu cầu)
    - Các thuộc tính protected

Phong cách lập trình

- Bao gồm các yếu tố:
  - Cách đặt tên hàm, biến
  - Cách xây dựng câu lệnh, cấu trúc chương trình
  - Cách viết chú thích

➔Hướng tới phong cách làm cho mã nguồn:

- Dễ hiểu, dễ sửa đổi
- An toàn (ít lỗi)

Người khác có thể hiểu được, bảo trì được

Tại sao cần dễ hiểu?

- Phần mềm luôn cần sửa đổi và nâng cấp

→ kéo dài tuổi thọ, nâng cao hiệu quả kinh tế

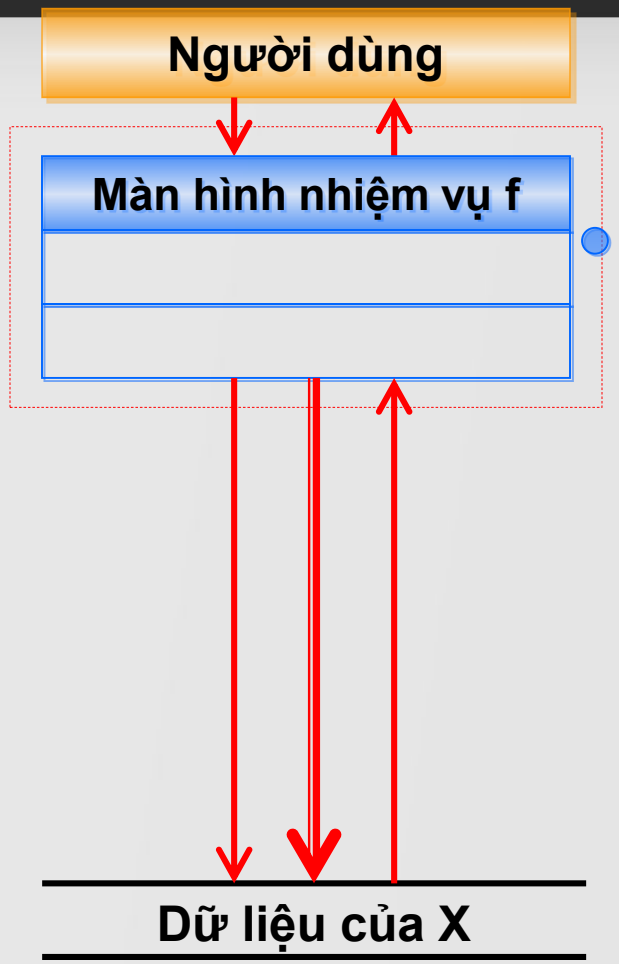
- Nếu không dễ hiểu:
  - Bảo trì tốn thời gian, chi phí cao
  - Phải bảo trì suốt vòng đời phần mềm
  - Bản thân tác giả cũng không hiểu

MÔ HÌNH ĐA LỚP (Multi-Layers)

Nội dung

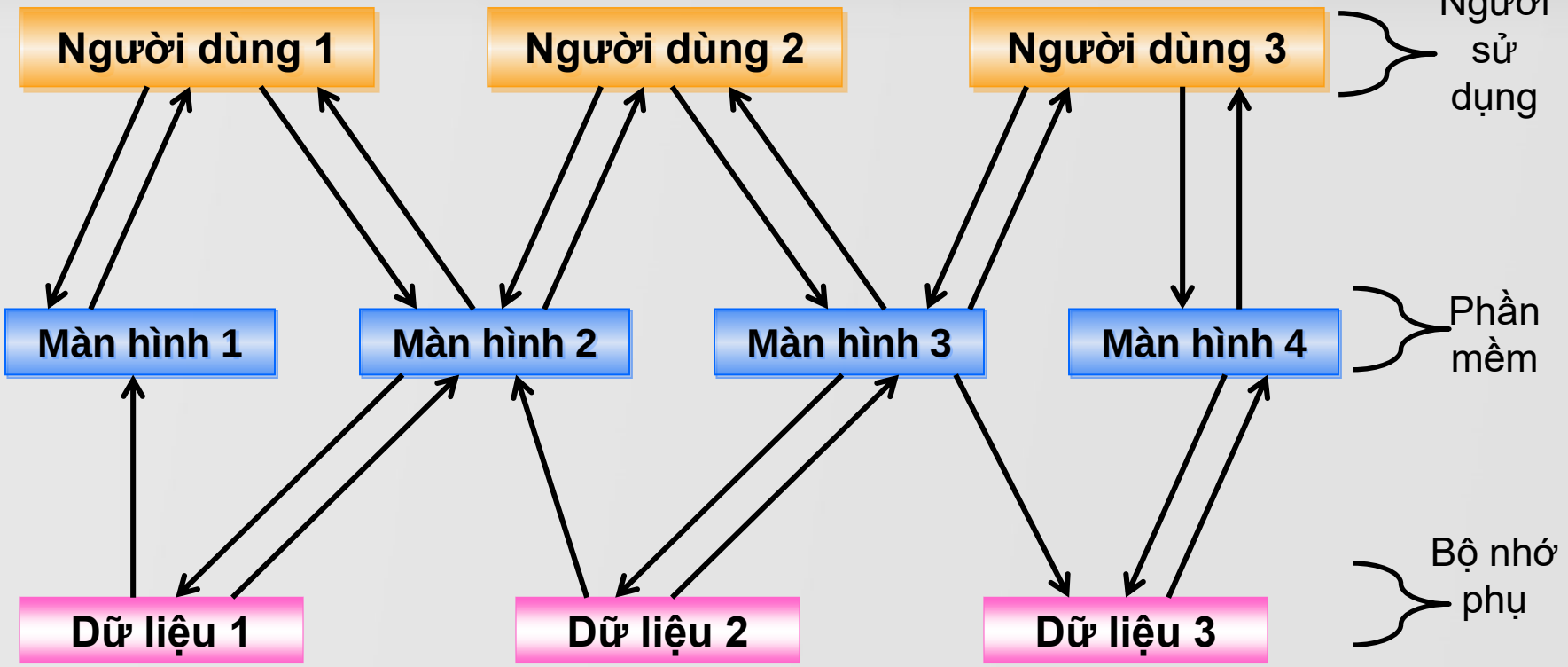
- Mô hình 1 tầng, 2 tầng, 3 tầng
- Phân biệt 3-tier, 3-layer
- Vai trò và nhiệm vụ của mỗi layer
- Quản lý ngoại lệ trong mô hình 3-layer

Mô hình kiến trúc 1 tầng (1 layer)



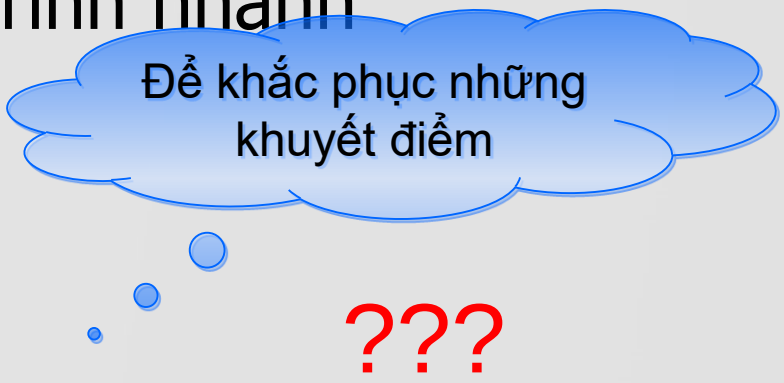
I/O
Xử lý biến cố
Xử lý chính
Xử lý trên dữ liệu

Mô hình kiến trúc 1 tầng (1 layer)

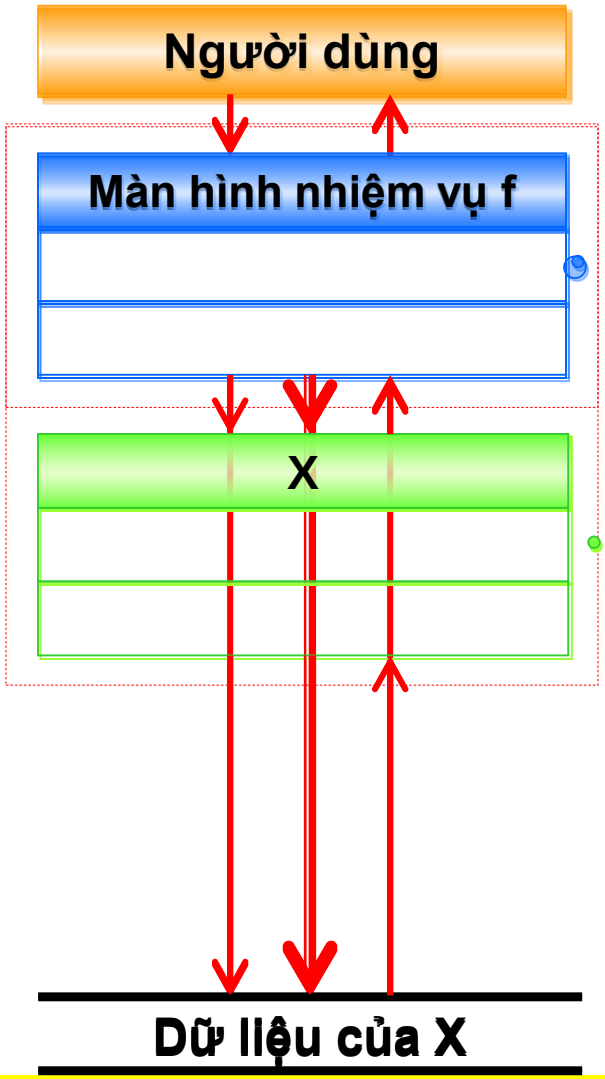


Mô hình kiến trúc 1 tầng (1 layer)

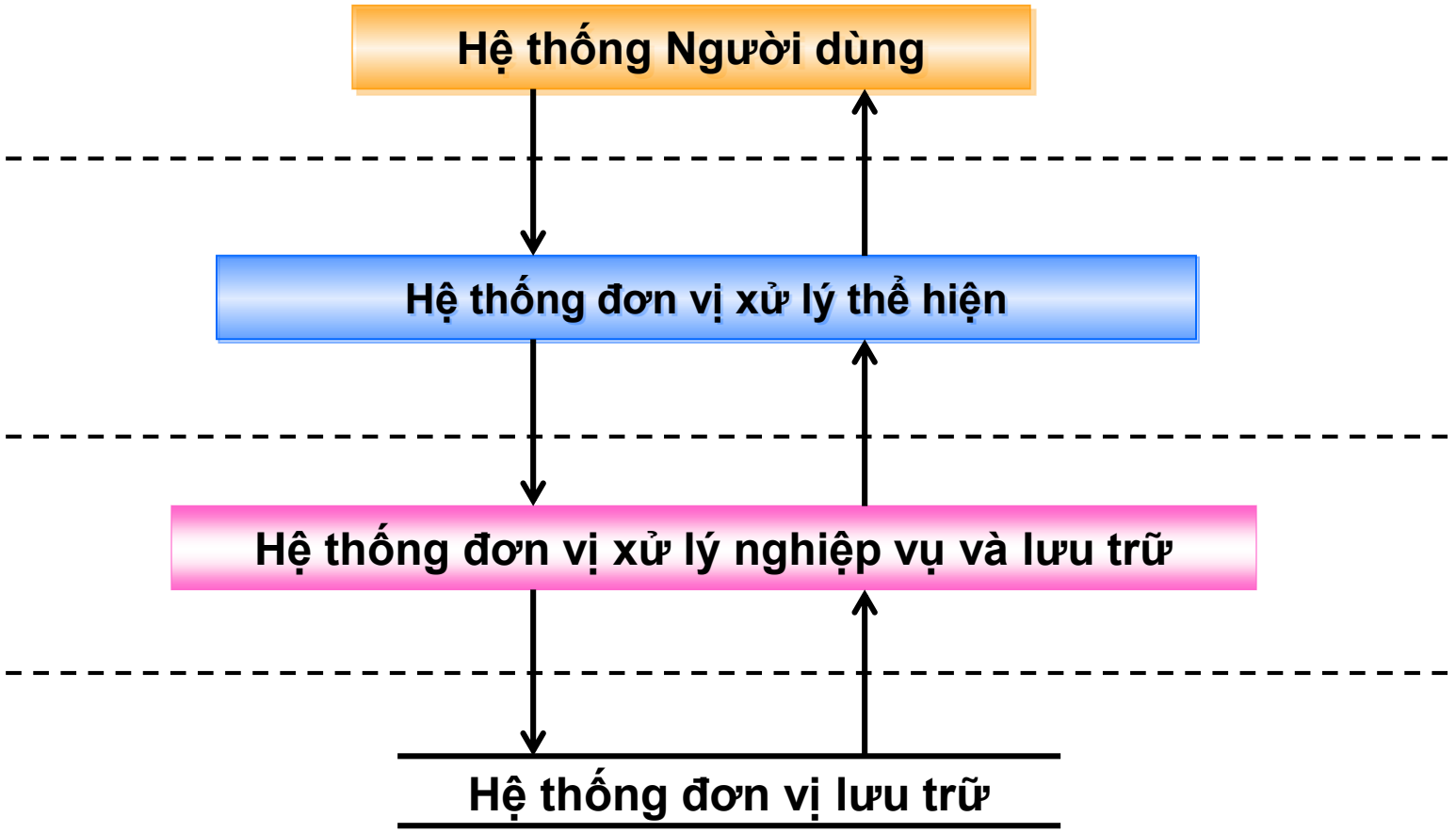
- Hệ thống trên bao gồm:
  - 3 người sử dụng
  - 4 đơn vị xử lý
  - 3 đơn vị lưu trữ
- Đặc điểm: Không có sự phân loại các xử lý
- Ưu điểm: Thiết kế và lập trình nhanh
- Khuyết điểm:
  - Mỗi đơn vị xử lý phức tạp
  - Khó bảo trì
  - Không có tính tái sử dụng



Mô hình kiến trúc 2 tầng (2 layer)



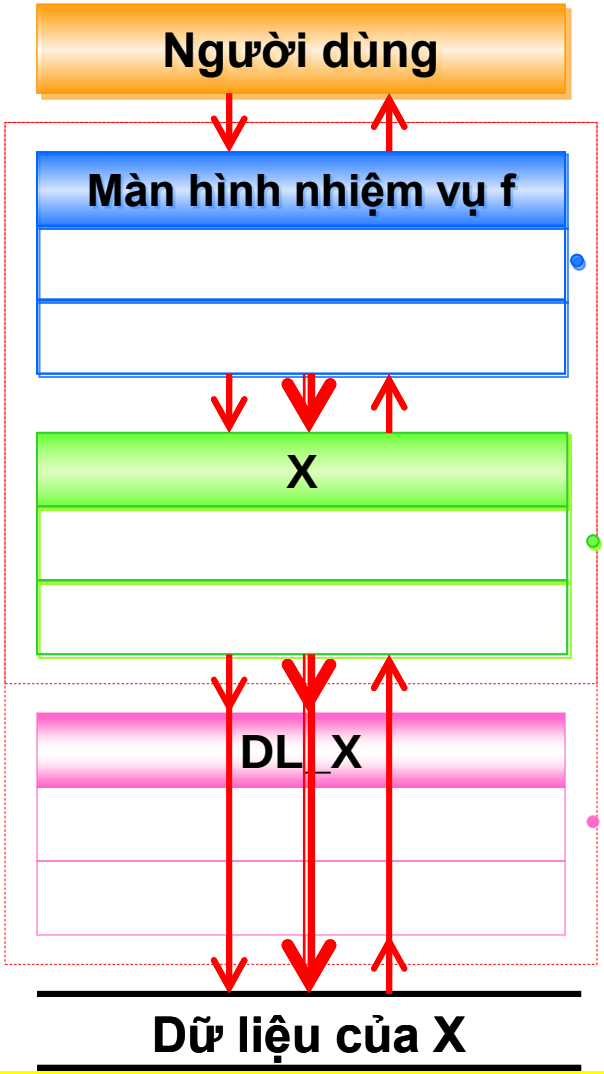
Mô hình kiến trúc 2 tầng (2 layer)



Mô hình kiến trúc 2 tầng (2 layer)

- Đặc điểm: Các đơn vị xử lý được phân thành 2 loại
  - Loại 1: Các đơn vị xử lý chuyên biệt về giao tiếp người dùng
  - Loại 2: Các đơn vị xử lý nghiệp vụ (kiểm tra, tính toán), lưu trữ (đọc, ghi)
- Ưu điểm, khuyết điểm ?

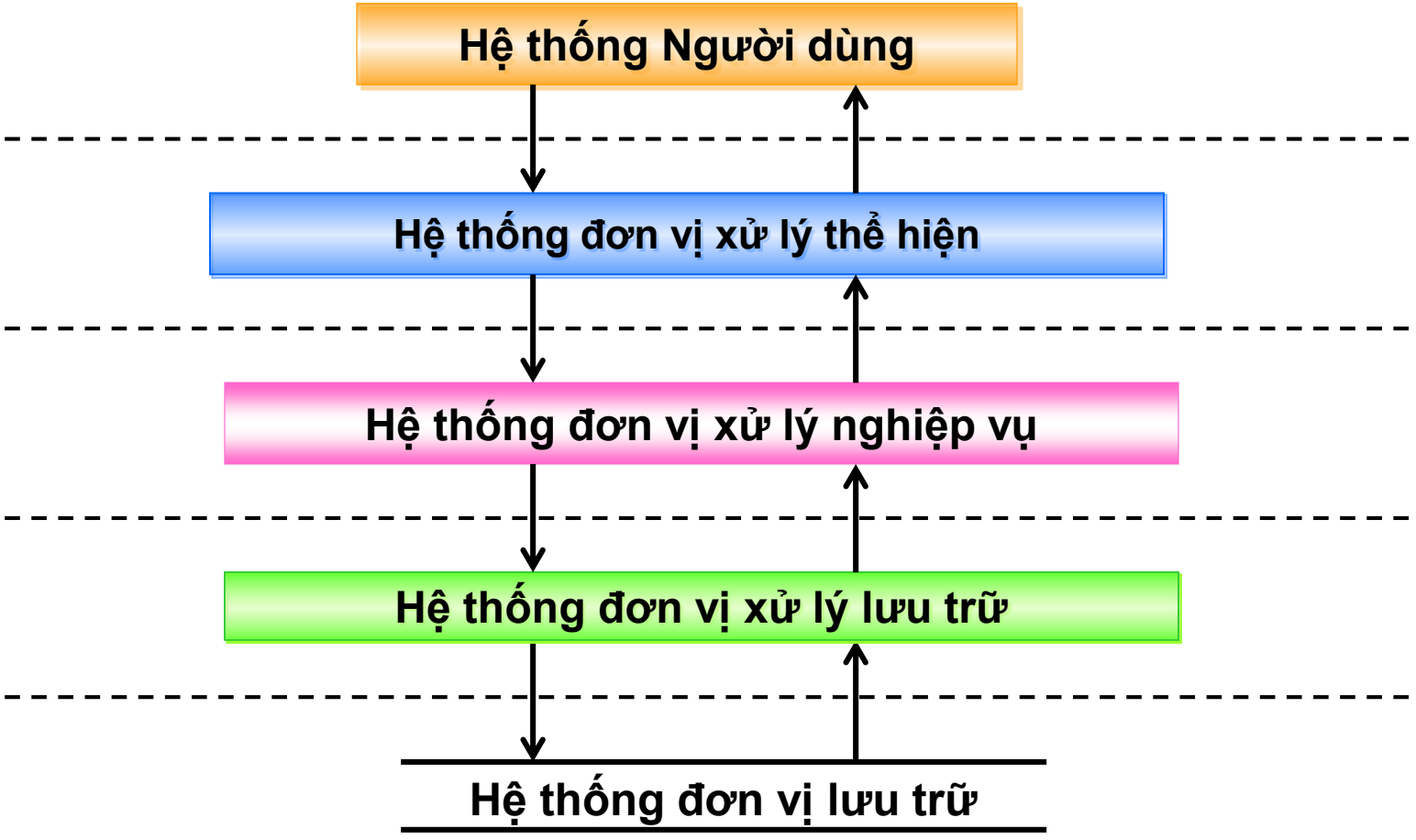
Mô hình kiến trúc 3 tầng (3 layer)



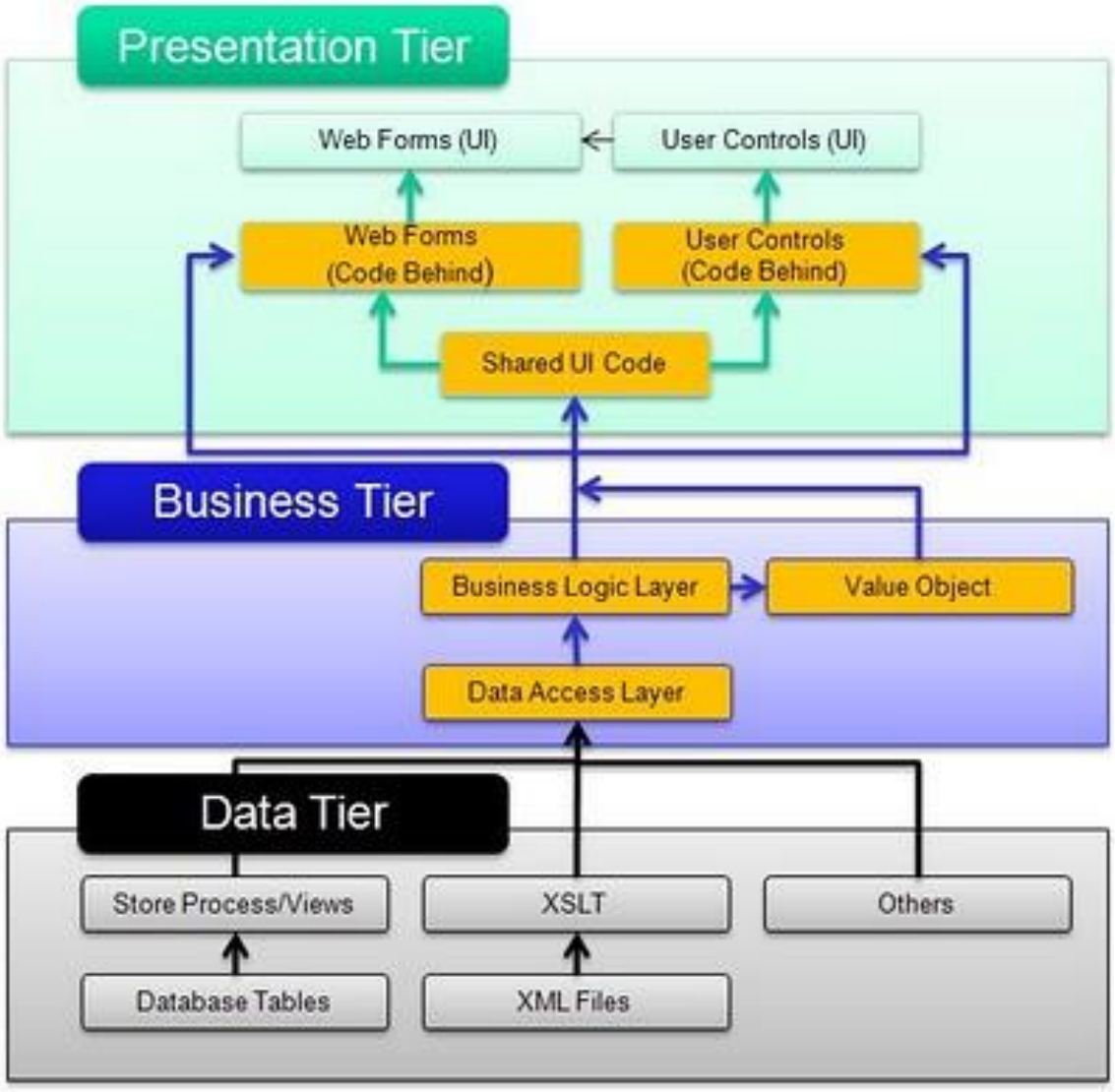
I/O
Xử lý biến cố

Xử lý chính
Xử lý trên dữ liệu

Mô hình kiến trúc 3 tầng (3 layer)



3-tier vs 3-layer



Vai trò của các layer

- GUI (Presentation) Layer:
  - Thu thập và hiển thị thông tin cho người dùng cuối.
  - Không sử dụng trực tiếp các dịch vụ của lớp Data Acces.
  - Sử dụng các dịch vụ do lớp Business Logic cung cấp.
  - Ở lớp này, chúng ta có thể bỏ qua các ràng buộc, các logic nghiệp vụ của ứng dụng.

Vai trò của các layer

- Business Logic Layer:
  - Lớp này thực hiện các nghiệp vụ chính của hệ thống (Ví dụ: kiểm tra các yêu cầu nghiệp vụ trước khi cập nhật dữ liệu)
  - Sử dụng các dịch vụ do lớp Data Access cung cấp.
  - Cung cấp các dịch vụ cho lớp Presentation

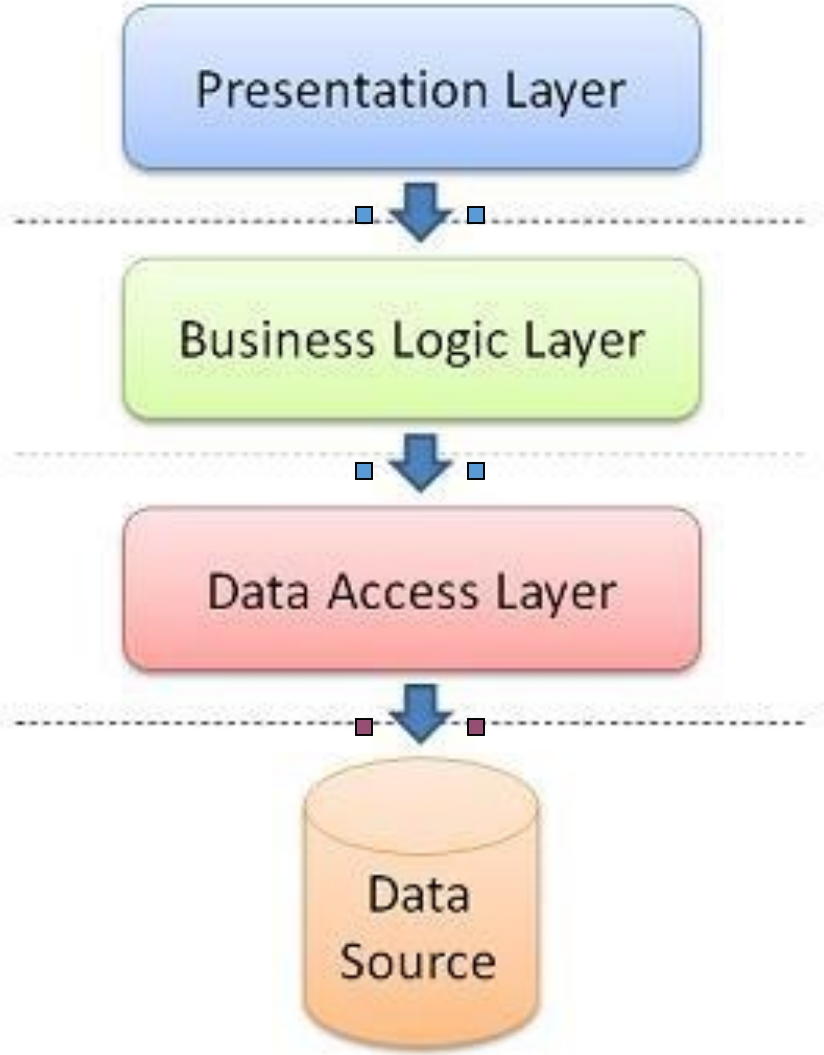
Vai trò của các layer

- Data Access Layer:
  - Lớp này thực hiện các công việc liên quan đến lưu trữ và truy xuất dữ liệu của ứng dụng.
  - Cung cấp các dịch vụ cho lớp Business Logic sử dụng.
  - Sử dụng các dịch vụ của các hệ quản trị cơ sở dữ liệu như MySql, SQL Server, Oracle,… để thực hiện nhiệm vụ của mình.

Các lưu ý quan trọng

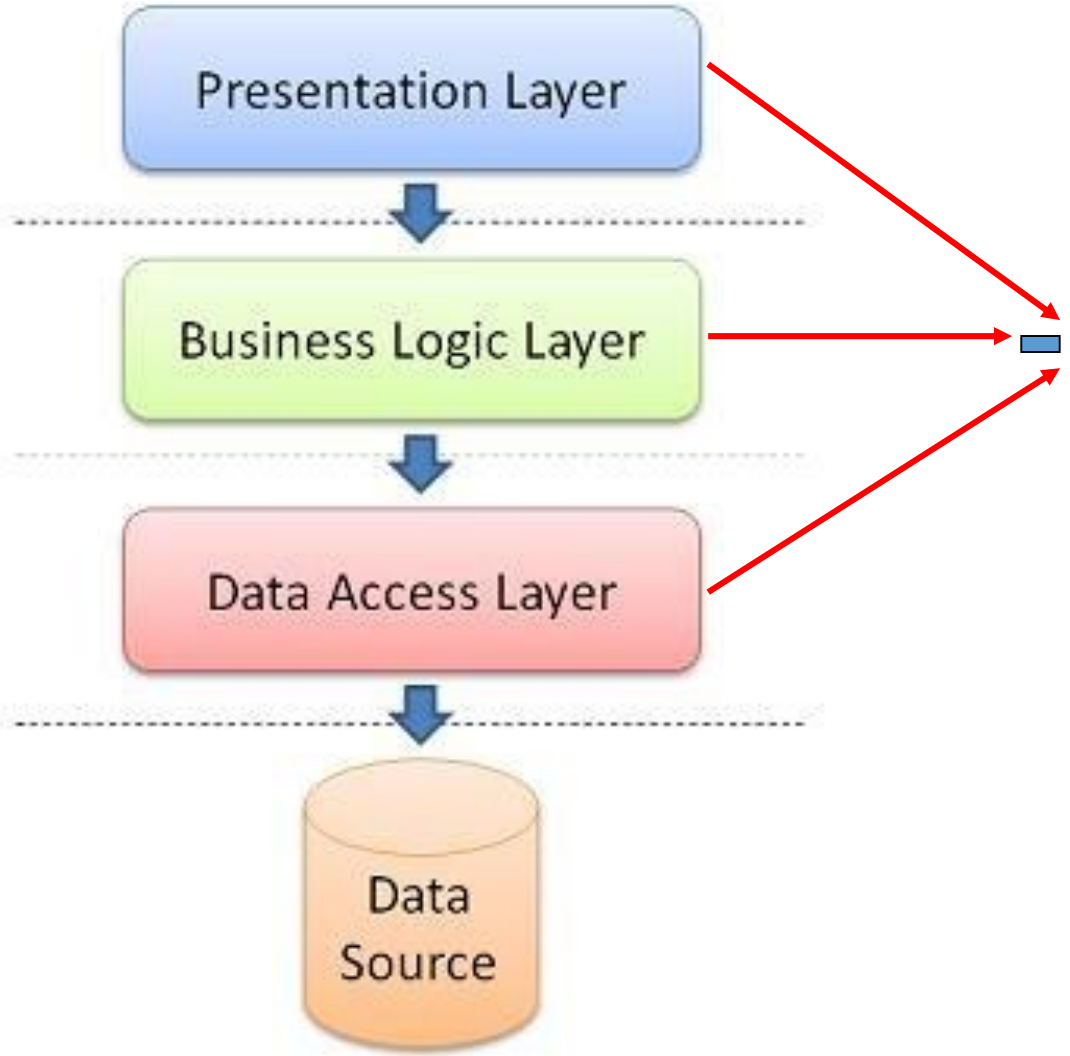
- Phân biệt vai trò Business Layer và khái niệm “xử lý”
- Mỗi Layer vẫn có xử lý riêng, đặc trưng của Layer đó
- Đôi khi việc quyết định 1 xử lý nằm ở layer nào chỉ mang tính chất tương đối

Việc trao đổi liên lạc giữa các layer



- **Data Transfer Object (DTO)**
- **Các giá trị, dòng, bảng**

Sự phụ thuộc giữa các layer



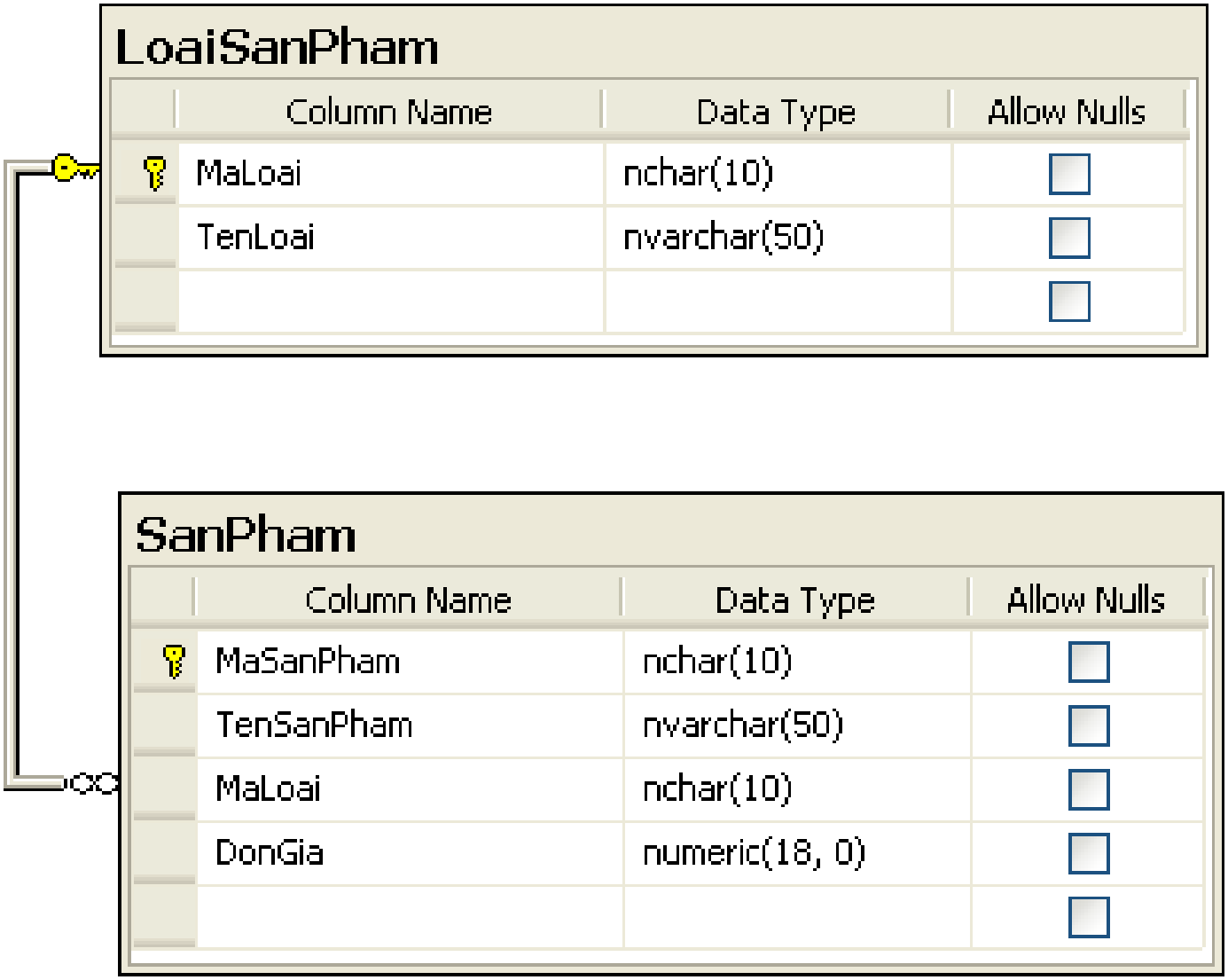
Tính chất của mô hình 3-layer

- Giảm sự kết dính giữa các thực thể phần mềm (decoupling)
- Tái sử dụng
- Chia sẻ trách nhiệm

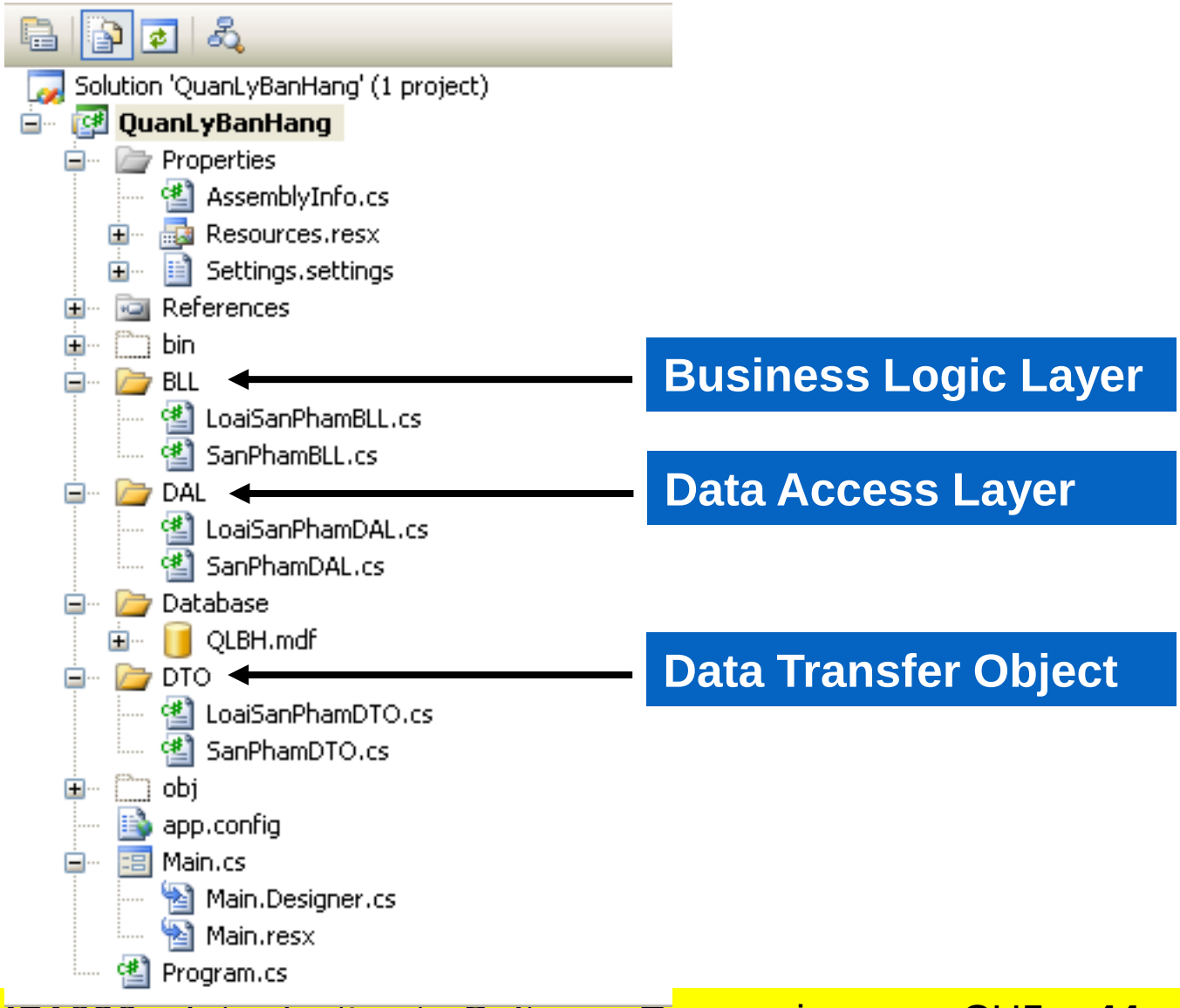
Quản lý ngoại lệ

- Ngoại lệ có thể xảy ra ở bất kỳ layer nào
- Khi ngoại lệ xảy ra ở một layer thì:
  - Xử lý nội bộ trong layer đó
  - "Quăng" ngoại lệ lên layer "cao hơn"
  - Không xử lý
- Khi một layer nhận ngoại lệ từ một layer "thấp hơn"
  - Xử lý nội bộ
  - "Quăng" ngoại lệ lên layer "cao hơn"
  - Không xử lý

Phát triển ứng dụng



Phát triển ứng dụng



Phát triển ứng dụng

- Các lớp DTO
  - Nội dung mỗi lớp gồm:
    - Fields
    - Các phương thức khởi tạo.
    - Các phương thức set, get
  - VD: SanPhamDTO, LoaiSanPhamDTO

Phát triển ứng dụng

- Các lớp DAL
  - Ứng với mỗi bảng trong database tạo một class DAL tương ứng.
  - VD: SanPhamDAL, LoaiSanPhamDAL

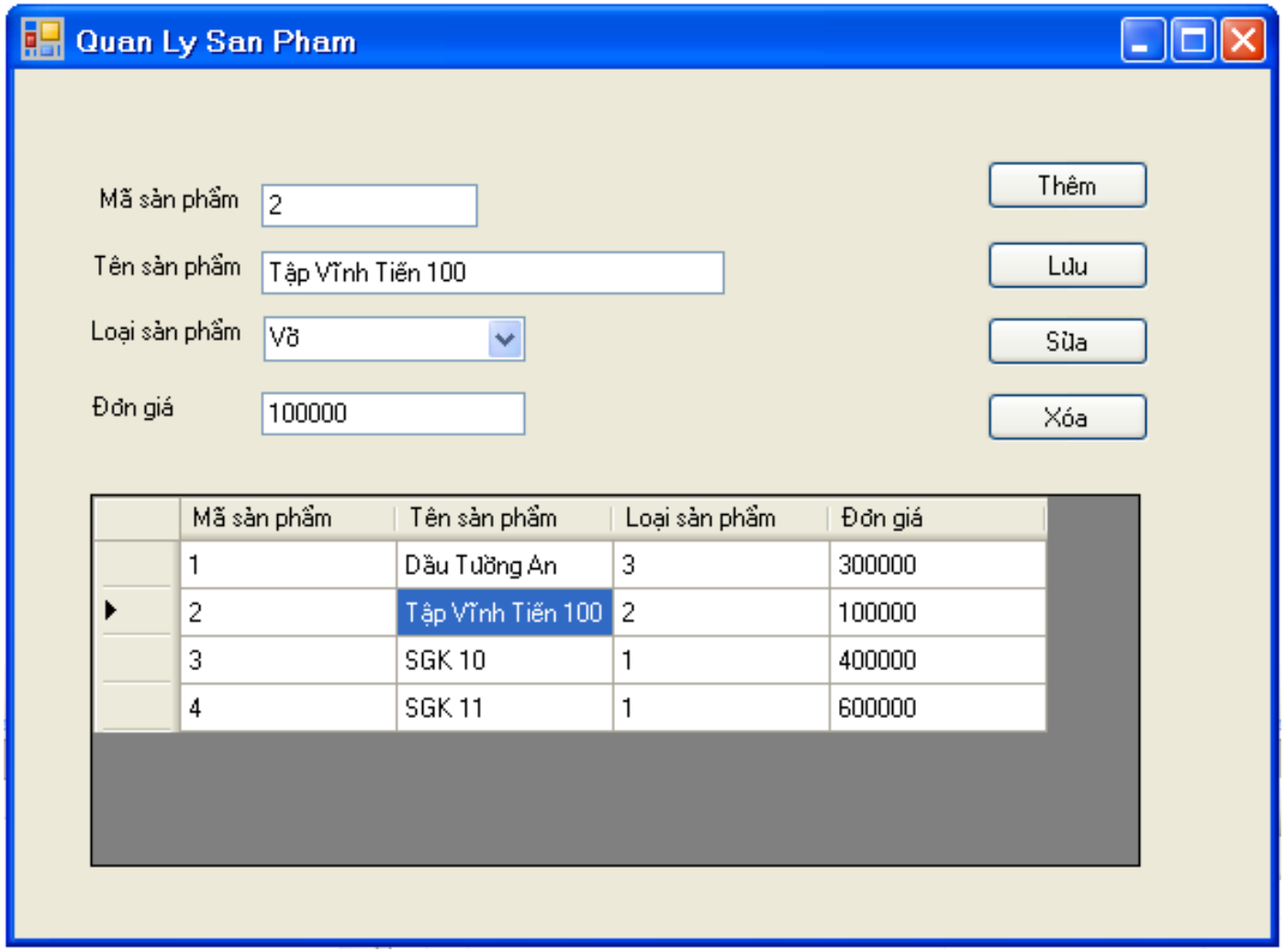
Phát triển ứng dụng

- Các lớp BLL
  - Tạo các class giao tiếp với lớp Presentation
  - Sử dụng các dịch vụ ở lớp DAL để xử lý nghiệp vụ.
  - VD: SanPhamBLL, LoaiSanPhamBLL

Phát triển ứng dụng

- Các lớp PL
  - Giao tiếp với người dùng.
  - Sử dụng các dịch vụ do lớp Business cung cấp.
  - VD: Xuất ra màn hình.

Phát triển ứng dụng



Phát triển ứng dụng

DEMO

Câu hỏi và thảo luận

Thank you!!!