ĐẠI HỌC CÔNG NGHIỆP TP.HCM
Khoa Công nghệ Thông tin

TỔNG QUAN
THIẾT KẾ PHẦN MỀM

TS. Huỳnh Hữu Nghĩa

huynhhuunghia@iuh.edu.vn

Nội dung:

- Khái quát
- Thiết kế dựa trên lớp logic.
- Các dịch vụ và các lớp
- Các bước thiết kế cho cấu trúc dựa trên lớp

Khái quát Thiết kế phần mềm

- Thảo luận cấu trúc tổng thể cho các ứng dụng trong việc phân nhóm logic các thành phần thành các lớp riêng biệt giao tiếp với nhau và với các client và ứng dụng khác.

- Liên quan đến phân chia logic và chức năng, và không tính đến vị trí vật lý của các thành phần.

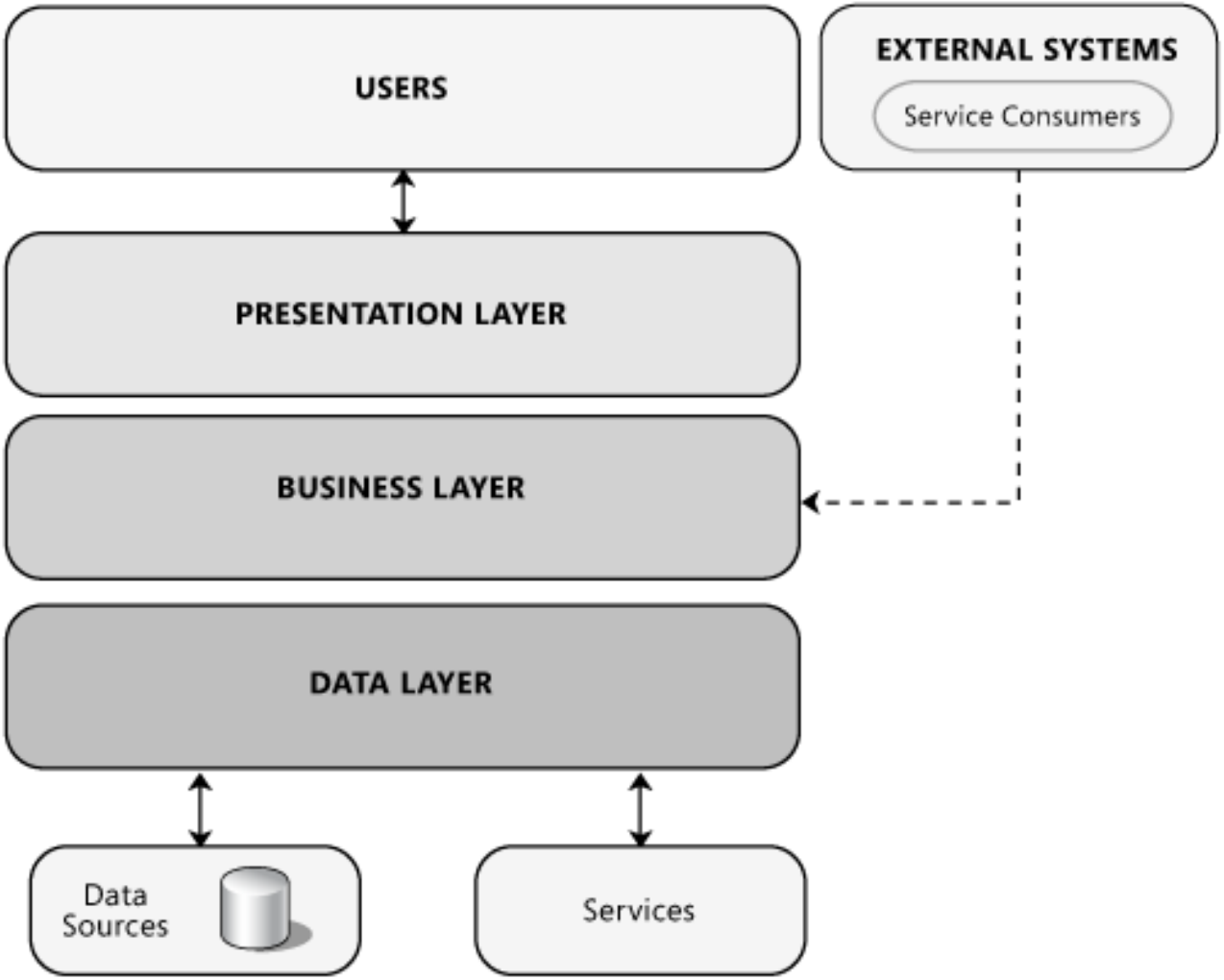
- Thực hiện phân chia logic, chọn cách bố cục chức năng phù hợp, cách ứng dụng hỗ trợ nhiều kiểu client.

Thiết kế dựa trên lớp logic

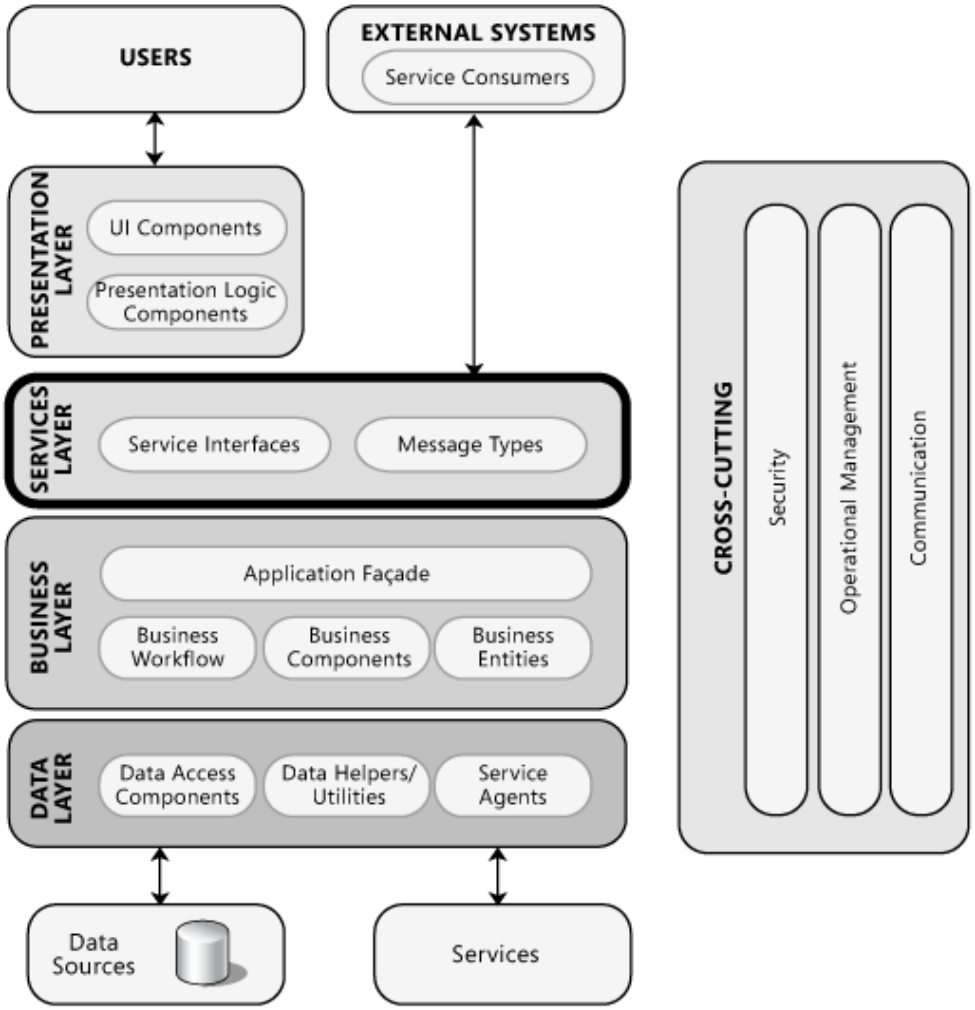
- Phân tách thiết kế thành các nhóm logic của các thành phần phần mềm. Các nhóm logic này gọi là các lớp.

- Mỗi lớp logic chứa một số loại thành phần riêng biệt được nhón thành các lớp con, mỗi lớp con thực hiện một loại công việc cụ thể.

Thiết kế tổng quát hệ thống



Các dịch vụ và lớp



Các bước thiết kế cấu trúc lớp

Bước 1: Chọn chiến lược phân lớp

Bước 2: Xác định các lớp yêu cầu

Bước 3: Quyết định cách phân phối các lớp và thành phần

Bước 4: Xác định nếu cần thu gọn các lớp

Bước 5: Xác định các quy tắc tương tác giữa các lớp

Bước 6: Xác định mối quan tâm xuyên suốt

Bước 7: Xác định giao diện giữa các lớp

Bước 8: Chọn chiến lược triển khai

Bước 9: Chọn các giao thức giao tiếp

Bước 1: **Chọn chiến lược phân lớp**

- Phân tách logic của các thành phần ứng dụng thành các nhóm thể hiện vai trò và chức năng riêng biệt.

- Phân chia ứng dụng quá ít hoặc quá nhiều lớp có thể làm tăng độ phức tạp không cần thiết; có thể làm giảm hiệu suất tổng thể, khả năng bảo trì, và tính linh hoạt.

Bước 1: **Chọn chiến lược phân lớp**

- Phân lớp hợp lý khi các lớp ứng dụng tương tác sẽ được triển khai trên cùng tầng và hoạt động trong cùng quy trình, cho phép tận dụng cơ chế giao tiếp hiệu quả cao hơn như gọi trực tiếp qua các giao diện thành phần.

- Để duy trì các lợi thế của phân lớp hợp lý và đảm bảo tính linh hoạt cho tương lai, cần phải duy trì đóng gói và khớp nối lỏng lẻo giữa các lớp.

Bước 1: **Chọn chiến lược phân lớp**

- Các lớp được triển khai ở các tầng riêng biệt, việc giao tiếp của các lớp liền kề sẽ xảy ra qua mạng, phải đảm bảo thiết kế hỗ trợ cơ chế giao tiếp phù hợp có tính đến độ trễ truyền thông và duy trì khớp nối lỏng lẻo giữa các lớp.

- Xác định những lớp nào có khả năng triển khai trên các tầng riêng biệt và cùng tầng để đảm bảo tính linh hoạt khi triển khai.

Bước 1: **Chọn chiến lược phân lớp**

- Cần xem xét đánh đổi khả năng sử dụng lại và khớp nối lỏng lẻo mà các lớp cung cấp chống lại tác động đối với hiệu suất và tăng độ phức tạp.

- Cẩn thận cách phân lớp ứng dụng và cách các lớp sẽ tương tác với nhau, đảm bảo sự cân bằng tốt giữa hiệu suất và tính linh hoạt.

Bước 2: **Xác định các lớp yêu cầu**

- Có nhiều cách khác nhau để nhóm chức năng liên quan thành các lớp, phổ biến nhất là tách chức năng ***trình bày***, ***dịch vụ***, ***nghiệp vụ*** và ***truy cập dữ liệu*** thành các lớp riêng biệt.

- Một số ứng dụng giới thiệu các lớp ***báo cáo***, ***quản lý*** hay ***cơ sở hạ tầng***.

Bước 2: **Xác định các lớp yêu cầu**

- Hãy cẩn thận khi thêm các lớp bổ sung, và không thêm khi chúng không cung cấp một nhóm hợp lý của các thành phần liên quan rõ ràng là *tăng khả năng duy trì*, *khả năng mở rộng*, hoặc *tính linh hoạt của ứng dụng*.

- Ví dụ, nếu ứng dụng không hiển thị các dịch vụ, lớp dịch vụ riêng biệt không thể được yêu cầu và có thể chỉ có các lớp ***trình bày***, ***nghiệp vụ*** và ***truy cập dữ liệu***.

Bước 3: **Phân phối các lớp**

- Phân phối các lớp và thành phần trên các tầng vật lý riêng biệt chỉ khi thật cần thiết như chính sách bảo mật, các ràng buộc vật lý, logic nghiệp vụ được chia sẽ và khả năng mở rộng.

- Trong ứng dụng Web, thành phần trình bày truy cập không đồng bộ thành phần nghiệp vụ, nên triển khai lớp nghiệp vụ và trình bày trên cùng tầng vật lý để tối đa hóa hiệu suất và dễ quản lý vận hành.

Bước 3: **Phân phối các lớp**

- Trong ứng dụng ***rich client***, nơi xử lý giao diện người dùng xảy ra trên máy tính để bàn, có thể triển khai thành phần nghiệp vụ trên một tầng riêng biệt vật lý cho lý do bảo mật và dễ dàng quản lý vận hành.

Bước 4: **Thu gọn các lớp**

- Ứng dụng có quy tắc nghiệp vụ rất giới hạn hoặc sử dụng các quy tắc chủ yếu để xác thực, có thể hiện thực cả logic *trình bày* và *nghiệp vụ* trong lớp đơn giản.

- Ứng dụng lấy dữ liệu từ dịch vụ Web và hiển thị dữ liệu, chỉ cần thêm một tham chiếu dịch vụ Web trực tiếp đến lớp trình bày và tiêu thụ dữ liệu dịch vụ Web trực tiếp, có thể kết hợp logic lớp *truy cập dữ liệu* và *trình bày*.

Bước 4: **Thu gọn các lớp**

- Một số ví dụ trên khi có thể thu gọn các lớp.

- Tuy nhiên, quy tắc phổ biến là nên luôn nhóm tính năng thành các lớp.

Bước 5: **Tương tác giữa các lớp**

- Xác định các quy tắc về cách mà các lớp sẽ tương tác với nhau nhằm giảm sự phụ thuộc và loại bỏ các tham chiếu vòng tròn.

- Một quy tắc phải tuân theo là chỉ cho phép tương tác một chiều giữa các lớp sử dụng một trong các hướng tiếp cận sau:

Bước 5: **Tương tác giữa các lớp**

- ***Tương tác trên-xuống***. Quy tắc này tránh phụ thuộc vòng tròn giữa các lớp, có thể sử dụng các sự kiện để làm cho các thành phần ở các lớp trên nhận biết thay đổi ở các lớp dưới mà không cần đưa ra các phụ thuộc.

Bước 5: **Tương tác giữa các lớp**

- ***Tương tác chặc chẽ***. Mỗi lớp chỉ tương tác lớp trực tiếp bên dưới. Lợi ích là sửa đổi giao diện của lớp chỉ ảnh hướng lớp trực tiếp bên dưới. Sử dụng hướng tiếp cận này khi thiết kế ứng dụng sẽ thay đổi theo thời gian, giảm thiểu sự tác động của những thay đổi hoặc có thể phân tán trên các tầng vật lý khác nhau.

Bước 5: **Tương tác giữa các lớp**

- ***Tương tác lỏng lẻo***. Các lớp mức cao có thể bỏ qua tương tác với các lớp mức thấp hơn trực tiếp. Điều này có thể cải thiện hiệu suất, nhưng sẽ tăng sự phụ thuộc. Hướng tiếp cận này chỉ được thiết kế cho ứng dụng không phân tán hay nhỏ.

Bước 6: **Mối quan tâm xuyên suốt**

- Sau khi xác định các lớp, cần phải xác định chức năng trải dài trên các lớp. Chức năng này gọi là mối quan tâm xuyên suốt gồm *nhật ký*, *bộ nhớ đệm*, *xác thực*, *ủy quyền* và *quản lý ngoại lệ*.

- Tránh trộn mã xuyên suốt với mã thành phần của lớp, các lớp và thành phần chỉ thực hiện cuộc gọi đến thành phần xuyên suốt khi phải thực hiện như ghi nhật ký, lưu trữ hoặc ủy quyền.

Bước 6: **Mối quan tâm xuyên suốt**

- Các thành phần xuyên suốt phải được triển khai để có thể truy cập được bởi tất cả các lớp – ngay cả khi các lớp nằm trên các tầng vật lý riêng biệt.

Bước 7: **Giao diện giữa các lớp**

- Xác định giao diện của một lớp, mục tiêu là thực thi khớp nối lỏng lẻo giữa các lớp. Một lớp không nên hiển thị chi tiết bên trong mà lớp khác có thể phụ thuộc.

- Giao diện của một lớp nên thiết kế để giảm các phụ thuộc bằng cách cung cấp giao diện chung mà ẩn đi chi tiết bên trong lớp. Việc che dấu này gọi là sự *trừu trượng* (*abstraction*).

Bước 7: **Giao diện giữa các lớp**

- Các hướng tiếp cận thiết kế như sau:
  - **Giao diện trừu tượng**. *Định nghĩa lớp cơ sở abstract hoặc mã lớp interface như định nghĩa kiểu cho các lớp cụ thể. Kiểu này là một giao diện chung mà tất cả người tiêu dùng sử dụng để tương tác với lớp.*
  - **Kiểu thiết kế chung**. *Mẫu thiết kế xác định các kiểu đối tượng cụ thể biểu diễn cho một giao diện vào các lớp khác nhau. Các kiểu đối tượng cung cấp một sự trừu tượng mà ẩn chi tiết có liên quan đến lớp.*

Bước 7: **Giao diện giữa các lớp**

- Các hướng tiếp cận thiết kế như sau:
  - **Dựa trên tin nhắn**. *Thay vì tương tác trực tiếp bằng cách gọi các phương thức hoặc truy cập thuộc tính của đối tượng, có thể sử dụng giao tiếp dựa trên tin nhắn. Một số giải pháp nhắn tin như: Windows Communication Foundation, Web service, và Microsoft Message Queuing.*

Bước 8: **Chiến lược triển khai**

- Tìm hiểu các mẫu triển khai phổ biến để chọn làm giải pháp triển khai tốt nhất cho ứng dụng.

(xem phần triển khai)

Bước 9: **Giao thức giao tiếp**

- Giao thức vật lý sử dụng cho giao tiếp qua các lớp hay tầng trong thiết kế đóng vai trò chính cho hiệu suất, bảo mật và độ tin cậy của ứng dụng.

- Lựa chọn giao thức giao tiếp thậm chí còn quan trọng hơn khi xem xét triển khai phân tán.


Q&A