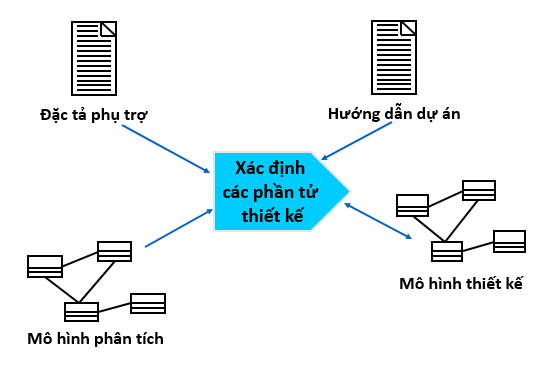
**I. XÁC ĐỊNH CÁC PHẦN TỬ THIẾT KẾ**

**1. Xác định các phần tử thiết kế**



- Input:

+ Thông số kỹ thuật, tài liệu kiến trúc phần mềm

+ Dự án hướng dẫn cụ thể

+ Mô hình phân tích

+ Mô hình thiết kế

- Output:

+ Lớp (Class)

+ Gói (Package)

+ Hệ thống con (Subsystem)

a. Mục đích

- Phân tích sự tương tác của các lớp phân tích và xác định các thành phần trong mô hình thiết kế

+ Lớp thiết kế (design class)

+ Hệ thống con (Subsystem)

+ Giao diện hệ thống con (Subsystem interface)

b. Tìm kiếm các lớp thiết kế

- Một lớp phân tích ánh xạ trực tiếp thành một lớp thiết kế nếu

+ Nó là một lớp đơn giản

+ Mức độ trừu tượng hóa đơn giản

- Các lớp phân tích phức tạp hơn có thể

+ Tách ra thành nhiều lớp

+ Trở thành một package

+ Trở thành một hệ thống con

+ Bất kỳ hình thức kết hợp nào

c. Nhóm các lớp thiết kế

- Nhóm các lớp dựa trên nhiều yếu tố:

+Phân bổ nguồn lực trong các đội phát triển

+ Tương ứng với từng loại người dùng

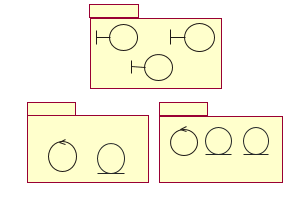
+Hệ thống con đại diện cho các sản phẩm và dịch vụ đã có mà hệ thống sử dụng

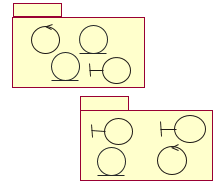
- Việc gộp nhóm hiệu quả giúp

+ Quản lý khả năng sử dụng lại

+ Bảo dưỡng hệ thống

d. Nhóm các lớp biên

- Nếu giao diện của hệ thống sẽ chắc chắn có các thay đổi đáng kể , các lớp biên được đặt trong các package riêng biệt.

- Nếu giao diện của hệ thống không chắc chắn có các thay đổi đáng kể, các lớp biên được gom nhóm với các lớp liên quan về chức năng.

e. Nhóm các lớp liên quan về mặt chức năng

- Các tiêu chí – liên quan về mặt chức năng:

+ Thay đổi/xóa bỏ một lớp làm ảnh hưởng tới các lớp khác

+ Hai đối tượng tương tác với nhau bằng một lượng lớn thông điệp hoặc có mối giao tiếp phức tạp

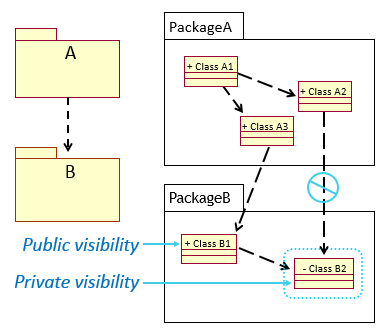
+ Lớp biên có thể có liên quan về mặt chức năng đến một lớp thực thể nào đó nếu lớp biên biểu diễn lớp thực thể đó

+ Hai lớp tương tác hoặc cùng bị ảnh hưởng bởi thay đổi trong cùng một tác nhân

+ Một lớp tạo ra thể hiện của lớp khác

- Các tiêu chí KHÔNG nên đặt hai lớp vào cùng một package:

+ Hai lớp liên quan đến các tác nhân khác nhau

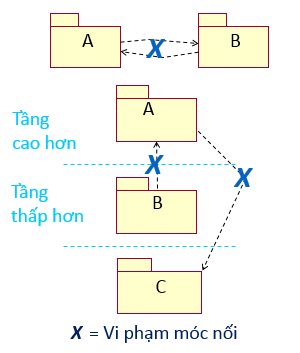
+ Một lớp bắt buộc và một lớp không bắt buộc

f. Sự phụ thuộc giữa các package

- Lớp public (+) có thể được tham chiếu bên ngoài package chứa nó.

- Lớp protected (#) chỉ được truy cập trong gói chứa nó và gói kế thừa gói chứa nó.

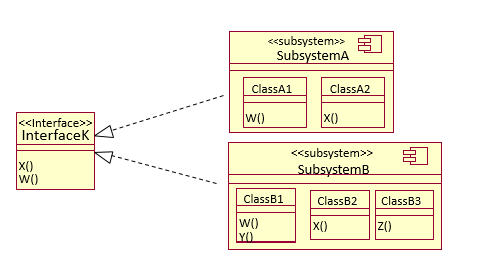
- Lớp private (-) chỉ được truy cập trong gói chứa nó.



- Các package không nên phụ thuộc lẫn nhau(cross-coupling)

- Package ở tầng thấp hơn không nên phụ thuộc vào các package ở tầng trên

- Nhìn chung, các phụ thuộc không nên bỏ qua các tầng ở giữa

VD: //////

**2. Hệ thống con (Subsystem)**

- Đóng gói hoàn chỉnh một hành vi nào đó

- Thể hiện khả năng độc lập sử dụng các giao diện một cách rõ ràng

- Có thể có nhiều hình thức thực thi

a. Sử dụng hệ thống con

- Phân chia hệ thống thành nhiều phần hoạt động tương đối độc lập

+ Thay đổi một phần không ảnh hưởng tới các phần còn lại

- Hệ thống con trong mô hình thiết kế sẽ trở thành thành phần trong quá trình cài đặt (components)

- Hệ thống con có thể được sử dụng để thể hiện một sản phẩm có sẵn, hoặc một hệ thống ngoại vi trong quá trình thiết kế

b. Tìm kiếm hệ thống con

- Các phân tích có thể trở thành hệ thống con nếu:

+ Cung cấp chức năng phức tạp

+ Các lớp biên (giao diện với hệ thống bên ngoài)

- Các sản phẩm có sẵn hoặc các hệ thống bên ngoài trong quá trình thiết kế (ví dụ thành phần):

+ Phần mềm giao tiếp

+ Hỗ trợ truy cập CSDL

+ Các kiểu và cấu trúc dữ liệu

+ Các tiện ích chung

+ Các sản phẩm theo ứng dụng

c. Gói và hệ thống con

- Hệ thống con

+ Cung cấp hành vi

+ Đóng gói hoàn chỉnh nội dung

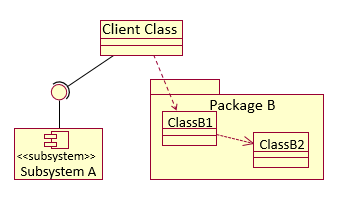
+ Dễ dàng thay thế

- Gói

+ Không cung cấp hành vi

+ Không hoàn toàn đóng gói nội dung

+ Có thể không dễ dàng thay thế



d. Giao diện cho hệ thống con

- Mỗi hệ thống con nên có một hoặc nhiều giao diện

- Mô hình hóa các giao diện

+ Ánh xạ giao diện vào hệ thống con

+ Chỉ ra sự phụ thuộc của nó tới các lớp khác

+ Chỉ ra các hành động của giao diện

• Tham số và kết quả

• Kiểu dữ liệu

- Đóng gói các giao diện

**3. Tính tái sử dụng**

a. Mục đích

- Sử dụng các giao diện để tìm cách sử dụng lại các hệ thống con hoặc các thành phần sẵn có trong hệ thống

b. Hướng dẫn

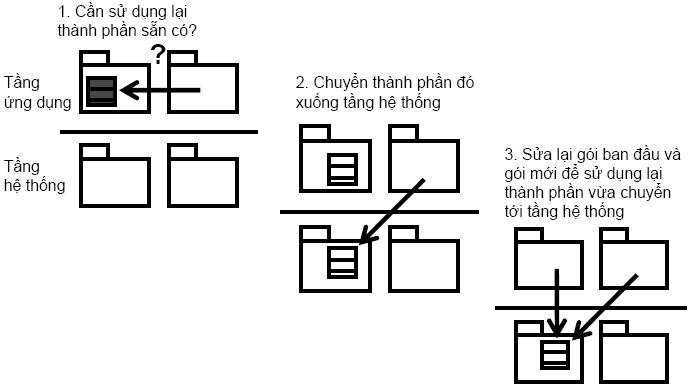
- Tìm kiếm các gần giao diện giống nhau

- Sửa giao diện cho phù hợp với giao diện sẽ sử dụng lại

- Thay thế giao diện có khả năng sử dụng lại với giao diện sẵn có (sử dụng lại)

- Ánh xạ hệ thống con của giao diện vừa bị thay thế vào thành phần có sẵn đó để sử dụng lại

c. Các khả năng sử dụng lại

- Bên trong hệ thống:

Tìm ra những điểm chung giữa các gói hoặc hệ thống con

- Bên ngoài hệ thống:

+ Sử dụng các thành phần sẵn có (thương mại, miễn phí).

+ Thành phần từ hệ thống phát triển trước đây.

+ Phát triển lại một thành phần có sẵn (sử dụng lại thiết kế).