**C1: Thông thường nên vẽ Activity Diagram trước hay Sequence Diagram**

Câu hỏi của bạn rất hay và liên quan đến quy trình phân tích và thiết kế phần mềm. Cả activity diagram (sơ đồ hoạt động) và sequence diagram (sơ đồ tuần tự) đều có vai trò quan trọng, nhưng thông thường nên vẽ theo trình tự sau:

**Thông thường nên vẽ Activity Diagram trước, rồi mới đến Sequence Diagram**

**Lý do:**

1. **Activity diagram mô tả quy trình tổng thể**
   * Activity diagram giúp bạn nắm bắt luồng công việc và logic nghiệp vụ ở mức cao
   * Giúp xác định các bước chính trong quy trình và các điều kiện quyết định
   * Cung cấp cái nhìn tổng quan về hệ thống trước khi đi vào chi tiết
2. **Sequence diagram đi vào chi tiết tương tác**
   * Sau khi đã hiểu rõ quy trình tổng thể, sequence diagram sẽ mô tả cụ thể cách các đối tượng tương tác với nhau
   * Xác định rõ các thông điệp được trao đổi giữa các đối tượng theo thứ tự thời gian
   * Yêu cầu bạn đã biết các thành phần/đối tượng trong hệ thống
3. **Quy trình phát triển hợp lý**
   * Từ yêu cầu → use case → activity diagram → sequence diagram
   * Activity diagram giúp hiểu "Cái gì" cần được thực hiện
   * Sequence diagram sau đó mô tả "Làm thế nào" để thực hiện nó

**Ngoại lệ:**

Có một số trường hợp có thể vẽ sequence diagram trước:

* Khi bạn đã rất quen thuộc với hệ thống và biết rõ các thành phần
* Khi bạn đang phát triển dựa trên một hệ thống có sẵn với kiến trúc đã được xác định
* Khi bạn cần tập trung vào mô tả giao tiếp giữa các thành phần cụ thể

**Tóm lại:**

Theo thực tiễn tốt nhất trong quy trình phát triển phần mềm, thứ tự vẽ nên là:

1. Activity diagram: Để nắm bắt luồng logic nghiệp vụ tổng thể
2. Sequence diagram: Để mô tả chi tiết cách thực hiện các tương tác

Cách tiếp cận này giúp bạn đi từ tổng quan đến chi tiết một cách hợp lý và đảm bảo các sơ đồ của bạn nhất quán với nhau.

**C2: generalization (kế thừa)**

1. Ý nghĩa của generalization trong use case:

* Thể hiện mối quan hệ "is a kind of" (là một dạng của)
* Use case con kế thừa và chuyên biệt hóa hành vi của use case cha
* Use case con có thể thêm các hành vi riêng biệt

1. So với <<extend>>:

* <<extend>> thể hiện tính năng mở rộng tùy chọn
* Generalization thể hiện rõ hơn việc đây là các CÁCH đăng nhập khác nhau
* Generalization phản ánh đúng hơn bản chất "is a" của mối quan hệ này