*Design pattern: Nói đơn giản thì nó là cách phối hợp giữa các class, các thành phần sao cho hợp lý nhất, và hệ thống sau này dễ mơ rộng. Khi gặp vấn đề, hãy nghĩ về pattern, ví dụ, khởi tạo thì có thể dùng factory. Làm việc với remote object có thể nghĩ tới proxy. Các phương pháp này đã dược test kỹ lưỡng, vì vậy nó giảm độ rủi ro., và nó cũng dự đoán luôn trong tương lai mình có khả năng sẽ thay đổi cái gì trong code, và làm cho sự thay đổi này dễ dàng nhất có thể.  
  
Thứ 2, trong các hệ thống, không chỉ có 1 pattern, mà rất nhiều pattern sẽ được áp dụng, sẽ dẫn tới chuyện, có thằng kết hợp pattern theo kiểu thế này, có thằng kết hợp theo kiểu khác. Và rủi ro xảy ra ở đây. Cho nên người ta chuẩn hóa cách kết hợp giwuax các pattern luôn. Không phải pattern muốn kết hợp với nhau thế nào cũng được, mà phải theo quy tắc. Ví dụ: Bộ 3 pattern đi chung với nhau, strategy, observer và composite pattern tạo thành kiến trúc Model-View-Control. Cái này gọi là Architect pattern.   
  
Còn domain driven là một khái niệm rộng hơn, nó giống như là 1 quy trình phát triển phần mềm. nó bao gồm domain + model và architect. Phần Domain và Model nó khá tương tự như OOAD, Nói tới domain có nghĩa là nói tới viêc làm việc với khách hàng để có requirement list. Phần Model tức là mô hình hóa nó lên. Việc mô hình hóa này cũng giống 1 bước trong OOAD, tức là dùng UML như usecase diagram để đưa ra mô hình về những việc cần làm, khách hàng có thể nhìn vào usecase này để biết xem yêu cầu cua rminhf có đúng không.... Và Cuối cùng là từ UML, requirement lists, ... Để nghiên cứu đưa ra kiến trúc cho hệ thống. Vẫn sử dụng architect pattern đã nêu ở trên.*